Diplomarbeit

ILR-RFS DA 21-57

Mixture Ratio and Combustion Chamber Pressure Control of an Expander-Bleed Rocket Engine with Reinforcement Learning

Karina Einicke

DIPLOMINGENIEUR

(Dipl.-Ing.)

Betreuer:	Dr. Christian Bach
Verantwortlicher Hochschullehrer:	Prof. Dr. techn. Martin Tajmar
Tag der Einreichung:	20.04.2021
Erster Gutachter:	Prof. Dr. techn. Martin Tajmar
Zweiter Gutachter:	Dr. Christian Bach

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen Institut für Luft- und Raumfahrttechnik

Professur für Raumfahrtsysteme

Aufgabenstellung für Diplomarbeit

ILR-RFS DA 21-57

Studiengang:	Maschinenwesen
Studienrichtung:	Luft- und Raumfahrttechnik
Name des Studierenden:	**Karina Einicke**
Matrikelnummer:	4504373

Thema: Mischungsverhältnis- und Brennkammerdruckregelung eines Expander-Bleed Raketentriebwerks mit Reinforcement Learning

Subject: Mixture Ratio and Combustion Chamber Pressure Control of an Expander-Bleed Rocket Engine with Reinforcement Learning

Motivation:

Im Rahmen des LUMEN Projektes des DLR am Standort Lampoldshausen können die Systemdynamiken des gesamten Raketentriebwerks untersucht werden. LUMEN ist ein Triebwerksdemonstartor, der aktuell am DLR entwickelt wird und ab 2022 getestet werden soll. Das Triebwerk verwendet eine neue Treibstoffkombination (LOX/Methan) und einen Expander-Bleed Zyklus. Mit seinen bis zu 6 Regelventilen ist LUMEN ein idealer Demonstrator, um den Einsatz und die Vorteile von fortgeschrittenen Regelalgorithmen an einem realen Triebwerksdemonstrator zu testen.
Die Möglichkeit, Systemdynamiken innerhalb einer immer besser validierten Simulationsumgebung (EcoSimPro/ESPSS) zu untersuchen, ermöglicht es weiterhin, „datenbasierende" Verfahren für die Triebwerkssteuerung und Regelung einzusetzen. Diese Verfahren, welche von den rasanten Entwicklungen in den Bereichen Maschinelles Lernen sowie Künstlicher Intelligenz profitieren, eigenen sich besonders für die Analyse und Regelung von komplexen, nichtlinearen Systemen. Im Rahmen der Diplomarbeit soll eine Triebwerksregelung für den komplexen Triebwerkszyklus des LUMEN Triebwerksdemonstrators erarbeitet werden. Durch den flexiblen Expander-Bleed Zyklus sowie bis zu 6 Regelventile bietet der Demonstrator vielfältige Möglichkeiten zur adaptiven Regelung unter Einsatz von Reinforcement Learning.

Aufgaben:

- Einarbeitung in das Thema der Steuerung und Regelung von Flüssigtreibstoff-Raketentriebwerken
- Einarbeitung in den LUMEN-Triebwerksdemonstrator
- Einarbeitung in Reinforcement-Learning
- Ableitung und Validierung eines Modells in EcoSim
- Generierung von optimalen Ventilsequenzen für transiente Vorgänge (Betriebspunktwechsel) unter Einhaltung der Randbedingungen (Brennkammertemperatur und Turbinendrehzahl)
- Optimierung der Triebwerksregelung
- Evaluierung der Robustheit des RL Reglers
- Auswertung und Evaluation der generierten Ergebnisse

Rechtliche Bestimmungen:

Der Bearbeiter ist grundsätzlich nicht berechtigt, irgendwelche Arbeits- und Forschungsergebnisse, von denen er bei der Bearbeitung Kenntnis erhält, ohne Genehmigung des Betreuers dritten Personen zugänglich zu machen. Bezüglich erreichter Forschungsleistungen gilt das Gesetz über Urheberrecht und verwendete Schutzrechte (Bundesgesetzblatt I/ S. 1273, Urheberschutzgesetz vom 09.09.1965). Der Bearbeiter hat das Recht, seine Erkenntnisse zu veröffentlichen, soweit keine Erkenntnisse und Leistungen der betreuenden Institutionen eingeflossen sind. Die von der Studienrichtung erlassenen Richtlinien zur Anfertigung der Studienarbeit sowie die Prüfungsordnung sind zu beachten. Der TUD-Betreuer ist während der Bearbeitung in Form von mindestens 3 Konsultationen über den Status der Arbeit zu informieren. Die „Richtlinien zur Anfertigung von Studienarbeiten" der Professur, die „Diplomprüfungsordnung" der Fakultät sowie die „Richtlinien zur Sicherung guter wissenschaftlicher Praxis" der TU Dresden sind zu beachten.

Betreuer:	Dr.-Ing. Christian Bach, ILR, TU Dresden Kai Dresia, DLR
Gutachter:	Univ.-Prof. Dr. Martin Tajmar, ILR, TU Dresden Dr.-Ing. Christian Bach, ILR, TU Dresden
Ausgabe:	04.01.2021
Abgabe:	03.06.2021

Empfangsbestätigung des Studenten:
Ich bestätige hiermit, dass ich die Aufgabenstellung sowie die rechtlichen Bestimmungen und die Studien- und Prüfungsordnung gelesen und verstanden habe.

Prof. Dr. Martin Tajmar	Betreuer	Prof. Dr. Martin Tajmar	Karina Einicke
Verantw. Hochschullehrer	Professur Raumfahrtsysteme	Studienrichtungsleiter	Studierende(r)

Hiermit erkläre ich, dass ich die von mir dem Institut für Luft-und Raumfahrttechnik der Fakultät Maschinenwesen eingereichte Diplomarbeit zum Thema *Mischungsverhältnis- und Brennkammerdruckregelung eines Expander-Bleed Raketentriebwerks mit Reinforcement Learning (Mixture Ratio and Combustion Chamber Pressure Control of an Expander-Bleed Rocket Engine with Reinforcement Learning)* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Berlin, 20.04.2021

Karina Einicke

Contents

Nomenclature

Indices

0	Initial or Atmosphere
CC	Combustion Chamber
com	Command or commanded
e	Exit or Exhaust
equiv	Equivalent
f	Fuel
in	Inlet
o	Oxidizer
out	Outlet
RC	Regenerative Cooling (Channel)
t	Current Timestep
t+1	Next Timestep
th	Throat

Liquid Rocket Engines

Δp	Pressure Loss
δP	Pressure Drop across Valve
$\dot{m}$	Ejected Mass Flow Rate
ϵ	Expansion Ratio
η	Efficiency
γ	Specific Heat Ratio
ρ	Density

τ	Valve Opening Time Constant
ζ	Flow Resistance
A	Area
c	Effective Exhaust Velocity
c_D	Discharge Coefficient
C_F	Thrust Coefficient
C_v	Valve Flow Coefficient
F	Thrust
g	Gravity of Earth
H	Enthalpy
I_{sp}	Specific Impulse
K_v	Flow Factor
Ma	Mach Number
MR	Mixture Ratio
P	Power
p	Pressure
pos	(Valve) Position
Q	Flow Rate (in US Gallons per Minute)
q	Heat Flow
R	Gas Constant
S	Specific Gravity
T	Temperature
TDH	Actual Total Dynamic Head of Pump
v	Velocity

Reinforcement Learning

α	Learning Rate
α_{trade}	Trade-Off Coefficient

δ	Temporal Difference Error
ϵ	Scale Factor
$\mathbb{E}$	Expected Value
γ	Discount Factor
$\mathcal{B}$	Replay Buffer
$\mathcal{N}$	Noise
μ	Policy
$\mu_{\theta_{targ}}$	Target Policy
ϕ	Q-Function Parameters
ϕ_{targ}	Target Network Parameters
ρ	Polyak Hyperparameter
τ	Trajectory
θ	Policy Parameters
a	Action
B	Sample Batch
d	Terminal State
G	Return or Cumulative Reward
J	Expected Return
P	Probability Distribution
Q	Q-Function
r	Reward
rew	Reward
s	State
t	Time Step
V	Value Estimate of Result State
y	Target

Acronyms

CH_4	Methane
LCH_4	Liquid Methane
LH_2	Liquid Hydrogen
BPV	Bypass Valve
BRF	Bleed Reward Function
CCV	Coolant Control Valve
CP	Checkpoint
DDPG	Deep Deterministic Policy Gradient
DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center)
ESA	European Space Agency
ESPSS	European Space Propulsion System Simulation
FCV	Fuel Control Valve
FPOV	Fluel Preburner Oxidizer Valve
GPM	Gallons per Minute
HMS	Hybrid Multi-Start
ICS	Intelligent Control System
LEC	Life Extending Control
LMDE	Lunar Module Decent Engine
LNG	Liquid Natural Gas
LOX	Liquid Oxygen
LUMEN	Liquid Upper-stage deMonstrator ENgine
MDP	Markov Decision Process
MFR	Mass Flow Rate

MFV	Main Fuel Valve
MOV	Main Oxidizer Valve
MR	Mixture Ratio
MSBE	Mean Squared Bellman Error
MVC	Multivariable Control
OCV	Oxidizer Combustion Valve
OP	Operation Point
OPFV	Oxidizer Preburner Fuel Valve
OPOV	Oxidizer Preburner Oxidizer Valve
PI	Proportional Integral
RAV	Regenerative Cooling Channel non-Adjustable Valve
RCV	Regenerative Cooling Channel Control Valve
RL	Reinforcement Learning
RREC	Robust Rocket Engine Concept
SAC	Soft Actor-Critic
SSME	Space Shuttle Main Engine
TBV	Turbine Bypass Valve
TCV	Thrust Control Valve
TD	Temporal Differences
TD3	Twin Delayed DDPG
TFV	Turbine Fuel Valve
TOV	Turbine Oxidizer Valve
XCV	Mixer Control Valve

1. Introduction

1.1. Motivation

Liquid rocket engine control is indispensable for more complex space transportation missions as it enables throttling, stopping, and restarting the engine. The ability to reuse rocket engine components might enable cost-saving, which brings a competitive advantage. Precise engine control is a requirement for reusable rocket engines, which includes performance optimization, health monitoring, and reducing propellant consumption. [1, 2]. Vertical landing can only be achieved by precise engine and thrust vector control. The Merlin engine cluster allows SpaceX to vertically land and reuse its first stage, which is a breakthrough in the space industry [3]. A liquid rocket engine can be controlled by preset valve sequences, ensuring transition between preset operation points like the European Vulcain engine [4]. If the engine receives feedback from its system and can act accordingly to reach its setpoints, closed-loop control is applied [1]. For throttleable engines, multivariable control allows controlling combustion chamber pressure and mixture ratio of the engine at the same time. The Space Shuttle Main Engine was the first large-scale reusable engine. The engine was throttleable and used closed-loop control for combustion chamber pressure and mixture ratio. [5].
However, presetting valve control sequences, which are still primarily used, can cause difficulties as the system behavior changes for multiple flights of reusable engines, due to gear wearing, extreme thermo-mechanical loads, and time low-cycle fatigue [6]. Derivation from the predefined operational points may occur, as preset valve control sequences are not adjusted according to the changed engine condition. The valve settings cannot react to engine malfunction or guarantee the engine's health.
Suitable closed-loop control systems can react on the engine condition, adjusting valve positions to reach optimum performance. Besides classical PID-based solutions, model-based approaches and Reinforcement Learning (RL) are currently studied for closed-loop engine control [7]. Reinforcement Learning, a form of artificial intelligence, can learn an optimal control strategy by interacting with existing engine simulators without the need for constructing suitable state-space models. Furthermore, the trained RL controllers are computationally cheap to use compared to model-based approaches with online optimization. An agent is trained to set valve positions to reach predefined target parameters, such as combustion chamber pressure and mixture ratio, within the simulation model. The agent learns the rocket engine's complex behavior and can adjust valve positions when system behavior changes. Once the training is completed, target values can be reached with low computational effort. The engine can also be optimized to operate in its optimal operating range or to reduce propellant usage. [8]
Training an agent in Reinforcement Learning brings along challenges. If the simulation model

is not accurate enough, a sim-to-real gap occurs and the trained agent cannot be transferred onto a real-world engine. [9]
Within this thesis, an engine control for Liquid Upper-stage deMonstrator ENgine (LUMEN) with Reinforcement Learning is to be established, optimized and its robustness analyzed. Liquid Upper-stage deMonstrator ENgine (LUMEN) is an expander-bleed bread-board engine, designed by Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center) (DLR), using six controllable valves to reach its operation points.

1.2. Objectives and Approach

The main objective of this thesis is to reach predefined operational points for LUMEN by using reinforcement learning. Engine control takes boundary conditions and constraints into account and is optimized for minimum fuel consumption.
To build a foundation for the thesis the fundamentals of liquid rocket engines as well as historical background and control methods are described in chapter 2. LUMEN is introduced and analyzed in chapter 3 and the engine simulation with EcosimPro is validated. Chapter 4 provides an overview of reinforcement learning and its algorithms, which is then applied in Chapter 5. The controller is trained to reach a given setpoint, which is defined by the combustion chamber pressure, mixture ratio, and cooling channel mass flow rate. Operation point transition is discussed and implemented and the robustness of the reinforcement learning agent examined.

2. Fundamentals of Liquid Rocket Engines

Chemical rocket engines can be categorized into solid propellant, liquid propellant, and hybrid engines, which are a mixture of both. A chemical rocket engine generates thrust by converting chemical energy into kinetic energy. The chemical energy is stored in the propellants and converted with the highest efficiency possible. The most simple bi-propellant liquid rocket engine consists of two pressured supply tanks that feed the main combustion chamber of the engine through pipes. Valves serve as the control elements to regulate the propellant flow. The propellants are inserted into the combustion chamber at a predefined Mixture Ratio (MR)[1] and are then released as gaseous combustion products[2], which leave the engine through the nozzle. The MR has an influence on the thermodynamic properties of the combustion process, like combustion temperature T_{CC}, specific heat ratio γ and gas constant R. [4, 10]
When fuel and oxidizer are combined, a chemical reaction takes place in the combustion chamber. The exothermic reaction in the combustion chamber using CH_4 and LOX can be used as an example combustion process and results carbon dioxide and water as the reaction products: $CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O + Heat$.
The gas mixture is accelerated through the converging part of the Laval nozzle until $Ma = 1$ is reached in the throat area. Reaching the diverging part of the nozzle, the gas expands to supersonic speed and leaves the nozzle generating thrust. The magnitude of the thrust depends on the propellant's mass flow rate as well as the exit velocity vector of the exhaust gas. Exit pressure of the exhaust gas at the exit plane of the nozzle as well as the atmospheric pressure influence the thrust, which is described by equation 2.1. The exhaust velocity is described in equation 2.2, its theoretical maximum value can be reached when the exit pressure at the exit plane reaches 0, which can only be realized with infinite expansion. The performance of the rocket engine can be determined by the specific impulse (equation 2.3). The stoichiometric mixture ratio can be calculated according to equation 2.4. [1]

$$F = \dot{m} v_e + (p_e - p_0) A_e = \dot{m} c \tag{2.1}$$

$$v_e = \sqrt{\frac{2\gamma}{\gamma - 1} R T_{CC} \left[1 - \left(\frac{p_e}{p_{CC}} \right)^{\frac{\gamma - 1}{\gamma}} \right]} \tag{2.2}$$

$$I_{sp} = \frac{F}{\dot{m} g_0} \tag{2.3}$$

[1] The mixture ratio is defined as the ratio between the oxidizer and the fuel mass flows (equation 2.4).
[2] The combustion is either ignited externally or is spontaneous, which requires a hypergolic propellant.

$$MR = \frac{o}{f} = \frac{\dot{m}_o}{\dot{m}_f} \tag{2.4}$$

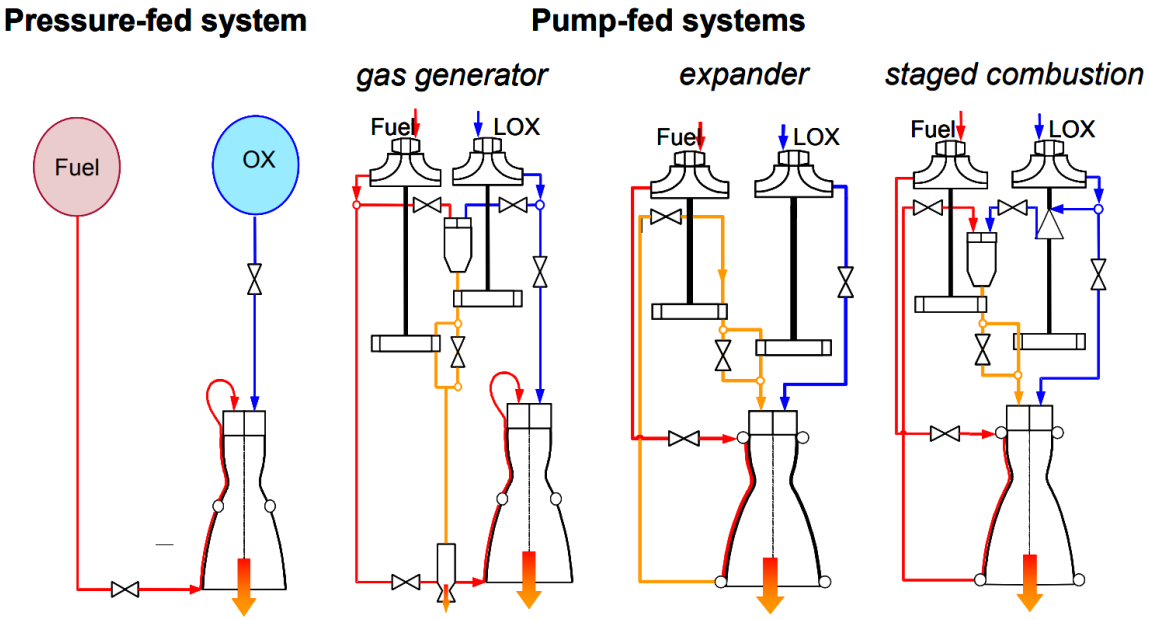


Figure 2.1.: Liquid Rocket Engine Cycles from Haidn [10]

The pressure-fed engine cycle is the simplest liquid rocket engine cycle. The combustion chamber pressure is relatively low because it only relies on the tank pressure to transport the propellants towards the combustion chamber. Greater combustion chamber pressure, and therefore thrust, can be generated by using a turbopump system, which can achieve greater pressure than tanks can provide to the combustion chamber. [4, 10]
In a gas generator cycle, some propellant is burned in a gas generator and the resulting hot gas is utilized to feed the turbines to generate power for the pumps. The gas generator cycle is considered an open cycle, as the burned gas is exhausted. When the propellant mass flow to the gas generator is increased, the turbine power increases, and the pumps can deliver more propellant into the combustion chamber, which increases the thrust. [4, 10]
The staged combustion cycle is a closed cycle in which the propellants are sent through multiple combustion chambers and therefore, burned in stages. One or multiple pre-burners combust a small amount of the propellants to power the pumps, which provide the main combustion chamber with propellants. To produce thrust the propellants are then burned in the main combustion chamber. The pre-burner exhaust flow can be injected into the main combustion chamber, so no gas is dumped and wasted. [4, 10]
A closed expander cycle is similar to the staged combustion cycle but misses the pre-burner. The fuel is heated up first, while it is used to cool the nozzle and combustion chamber walls. Vaporized it is fed into the turbine, which drives the pumps. The fuel is then injected into the combustion chamber along with the oxidizer at an optimal mixture ratio. A variation is an

expander-bleed cycle, which does not inject the entire fuel (used to drive the turbine) into the combustion chamber. [4, 5, 10]
In all cycles, the propellants can be used to cool the nozzle and the combustion chamber walls (Figure 2.1).

2.1. Control Loops

The main objective of rocket engine control is to reach predefined operating points in the combustion chamber (consisting of combustion chamber pressure and MR). Operating points can be reached by adjusting the control valves while operating constraints are met at all times. By manipulating combustion chamber pressure and MR thrust, specific impulse can be adjusted. [11]
Each type of rocket engine requires an individual control approach as the engine behavior highly depends on the engine design. Mechanical, structural, and thermal as well as chemical, electrical, and hydraulic aspects have to be taken into account and merged into one model. The system requirements, constraints, variable parameters, and behavior have to be considered to select an appropriate control system. The engine's performance (accuracy, response time, perturbation reliability, and rejection) relies on engine components such as flow-rate controllers and hydro-mechanic devices. Control loops can be sectioned into open-loop and closed-loop control. To control specific engine variables there are mainly three different control approaches: thrust-level, propellant-utilization, and thrust-vector control. To control the thrust p_{CC} and MR are central variables as well as the regulation of the tank pressure. If the engine involves a gas generator the number of influencing variables increases. [1, 11]

2.1.1. Open-Loop Control

Open-loop control measures variables with suitable instrumentation, but the engine itself does not take an action as a reaction to the measurements. This control method is therefore simple and preferred in space applications. Open-loop control is an option for conventional rocket engines, as their flight is naturally stable. It is however limited when high performance or robustness levels are requested since in those cases transient response of the system is necessary. The external conditions vary (due to e.g. altitude change) and even with constant valve positions the operating point is thus altered. Open-loop control can only offer a limited amount of thrust and/or MR control. [1]

2.1.2. Closed-Loop Control

Closed-loop control systems can adjust themselves or another system, according to the feedback it receives. The system uses sensors to measure predefined variables and gives commands to compensate for detected errors. Calibration is not required, but the computer needs to be able to take measures according to the output variables. On-off control is e.g. used for smaller spacecraft engines, where a single valve regulates tank pressure and can be switched to open or closed. [1, 11]

Thrust-Level Control

The thrust level is influenced by the injected mass flow into the combustion chamber and thus the combustion chamber pressure. Thrust control can be achieved by adjusting the propellant valve. Regulators or controllers can be employed for greater precision. More difficult approaches to influence the thrust are the variation of the throat area and the I_{sp} (equation 2.1). Changing MR modifies the I_{sp}, but can lead to declining performance. [1, 11]

Propellant Mixture Ratio and Propellant-Utilization Control

Propellant MR and propellant-utilization control is performed open-loop or closed-loop to achieve maximum I_{sp} and minimize propellant resources. Open-loop control can be extended by additional adjustable orifices to control the propellant flow. By adjusting the MOV, MR can be controlled. Restartable engines and high-velocity increment upper stages use closed-loop control. During start-up and shutdown MR can highly vary. To improve propellant management, the control system could move back the requirement of constant MR. Especially at the end of the mission, it is wise to empty the tanks to reduce the mass of the vehicle. [1, 11]

Thrust-Vector Control

Thrust-vector control is accomplished by a gimballed thrust chamber, gimballed nozzle, jet vanes in the nozzle, or a secondary injection into the main exhaust flow to guide the vehicle's direction. [11]

2.1.3. Reusable Liquid Rocket Engine Control

The propulsion system is the most unreliable system in space transportation. The liquid rocket engine system is complex and engine failure often results in the loss of the vehicle. The engine component is degraded as they are exposed to extreme thermo-mechanical loads during the flight. Thus, low-cycle fatigue damage and time-dependent damage (such as creep and material wear), are common degradation. Reusable rocket engines make the rocket or stage reusable, which is more cost-efficient than building new components for each flight. It reduces the cost of space flight. However, the maintenance of the components becomes more complex and expensive. On the other hand, the engine will be flight-proven after the first flight, and data about how the engine behaves during the flight can be used for the next missions. [6]
To be able to reuse an engine, health monitoring is crucial. Urgent failures (leakage) and slow failures (gear wearing) are detected and reported. Health monitoring does not only include fault detection, but also diagnosis, decision making, and malfunction control. Including health monitoring techniques in the propulsion system provides control capacity, which can prevent the destruction of components due to failures and ensure mission success. It has a high priority and is linked to engine control. [12]
Adjusting the rocket engine's performance during the flight, as well as protecting parts from

wearing out, are part of engine control. Controlling the engine to take care of engine parts is as important as adjusting the performance to reach different operating points to be able to reuse the engine. Thus, closed-loop control is usually applied. [5, 11]

Multivariable Control

Multivariable Control (MVC) is used in rocket engine control, when various valves can be adjusted to control p_{CC} and MR. It can provide more accurate control of the rocket engine than single-loop control. It usually relies on linear state space models, which are to be controlled. [5, 11]
MVC can be used to achieve fault tolerant and robust control. The selected variables for closed-loop control could be combustion chamber pressure and MR, which would provide engine throttling via setpoint control. When a staged combustion cycle is used, controlling outlet temperatures of the turbopumps are useful to regulate preburner combustion temperature. Not holding the turbopump temperatures at an optimum level may cause a decrease in turbine efficiency. Thus, closed-loop control of variables such as the turbopump outlet temperature along with combustion chamber pressure and MR are useful to hold the engine at maximum efficiency, while it is throttleable. [13]

Intelligent Control System

An Intelligent Control System (ICS) uses sensors or monitoring instruments to diagnose and predict engine behavior on-board. Sensors, actuators, and hardware failures can be detected and variables adjusted. ICS includes real-time engine diagnosis and prognosis, component condition monitoring, life-extending control, and adaptive control. Mission-level control provides requirements, such as thrust and MR, while propulsion-level control adjusts variables within the requirements to achieve thrust and MR and passes on commands to engine-level control. If difficulties (e.g. frozen valves) occur in the engine, new maximum reachable thrust depending on the new valve position can be determined. Engine-level control selects the best combination of engine settings according to its state. New values are transmitted to the propulsion-management system and by monitoring engine behavior (e.g. turbine discharge pressure) the engine's health can be evaluated. The engine's performance is reduced and the life span lengthened. Thrust and MR are adjusted, depending on the engine's health, performance (in terms of efficiency), and aging state of the components, even in a multi-engine system. [5, 11]

Life Extending Control

The key concept of Life Extending Control (LEC) is reducing damage (creep, fatigue, and fracture) of the rocket engine. Like any other concept of rocket engine control, LEC has to maintain performance requirements of the system, while primarily enhancing the durability of the system. Linear and non-linear control techniques are employed to control p_{CC} and MR. Minor performance losses can be accepted to reduce damage to lengthen the engine's life

span. [5, 11]
To further reduce damage during the critical transient phases (start-up and shut-down), LEC can be complemented by multidisciplinary optimization. Robust Rocket Engine Concept (RREC) addresses critical components, transient control parameters, and therefore, the whole operating cycle to minimize engine component damage. [11]

2.2. Control Valves

Valves are integrated into the rocket engine system to control the propellant flow, feed engine components, serve as safety elements to relief devices, and function as by-pass elements. The valve should require minimal flow force, minimal weight, and good sealing capability. The type of fluid, actuation energy, accuracy, opening, and closing speed need to be taken into account. [14, 15, 16]
The valve's actuation energy can be pneumatic, hydraulic, or electric. The most common approach to open and close valves is via pneumatic actuators. An electrically driven valve, named pilot valve, controls the actuator. The actuator is connected to a pressurized gas source or a venting line and is switched by the pilot valve, which is controlled by a chain of electrical components connected to the control computer (Figure 2.2). [15, 16]
Helium gas, which is expensive, is utilized to operate pneumatic-driven valves. By using electric-driven valves the throttling efficiency can be improved. Electric actuators have a control board, which terminates signals and power to the drive motor, which opens and closes the valve. [11]
In modern engines, electronically controlled actuators are used for more effective control. However, those tune-able valves require large bandwidths and show higher energy consumption. [4]

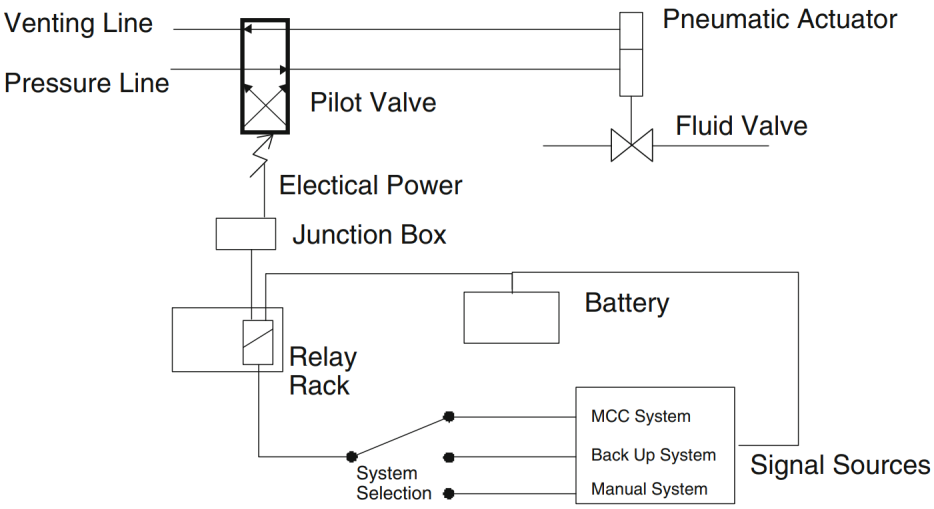


Figure 2.2.: Pneumaltic Actuator from Kitsche [15]

2.2.1. Flow Characteristics

The relation of flow coefficient and valve travel can be described by the flow characteristic of the valve. Depending on the opening position of the valve, it allows a certain amount of fluid to travel through the valve. Predicting the flow depending on the valve position enables flow regulation. The flow through the valve at any given opening position and the pressure differential are important to characterize the valve (Figure 2.3). [17]
The dimensionless flow resistance ζ is calculated according to equation 2.5, using the discharge coefficient c_D. [18]

$$\zeta = \frac{1}{c_D^2} \tag{2.5}$$

The pressure loss through the valve can be calculated according to equation 2.6.[1]

$$\Delta p = \zeta \frac{\rho}{2} v^2 \tag{2.6}$$

The flow coefficient C_v describes the relationship between the pressure drop and the flow rate through the valve. The valve flow coefficient is defined in equation 2.7, which is expressed in Gallons per Minute (GPM) at 60 °F. The fluid flow Q is measured in US GPM and S represents the specific gravity. The metric equivalent (K_v) can be calculated according to equation 2.8 (expressed in m^3/hr at 1 bar pressure loss at a temperature of 5 °C to 40 °C). [17]

$$C_v = Q\sqrt{\frac{S}{\Delta p}} \tag{2.7}$$

$$K_v = \frac{C_v}{1.15} \tag{2.8}$$

In a linear flow characteristic the relation between flow rate and valve position is linear. Equal percentage valves are usually applied in pressure control operations, expecting high variation in pressure drop. With an increasing valve opening the flow increases exponentially. [17]

2.2.2. Valve Types

Different valve types have different flow characteristics. Throttling and flow rate control, flow resistance when fully open, the opening, and closing mechanism, and tight shut off as well as preventing return flow and pre-set opening conditions have to be taken into account. [17]
The most common valves which can be used in liquid rocket engines can be seen in Table 2.1.

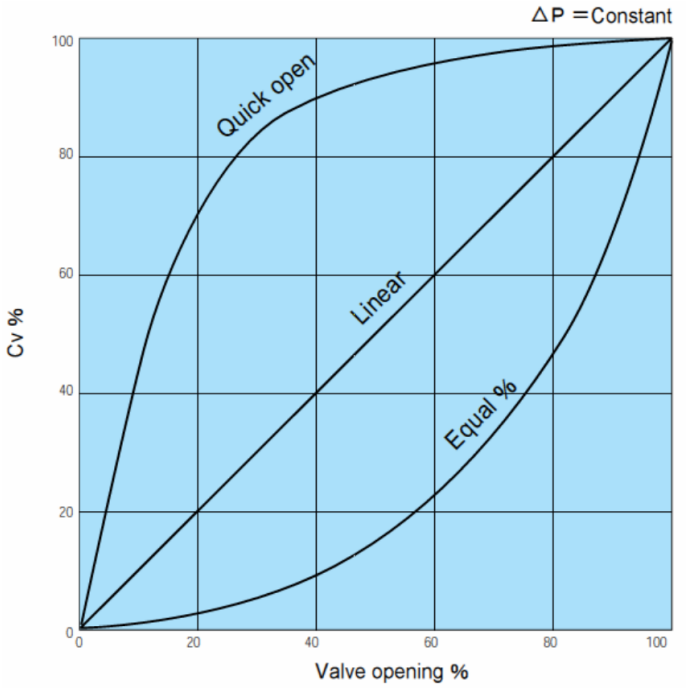
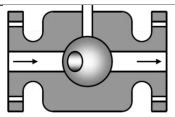
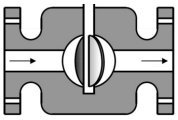
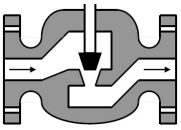
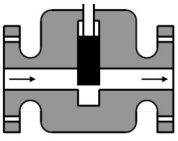


Figure 2.3.: Valve Flow Characteristics from Bhatia [17]

Table 2.1.: Valve Types from Bhatia [17] Gaphic Illustration from Reddy [19]

Valve	Flow Characteristic	Opening Conditions	Illustration
Ball Valve	quick opening linear	limited throttling fully open/closed	
Butterfly Valve	equal percentage linear	throttling fully open/closed	
Globe Valve	equal percentage linear	throttling precise regulation	
Gate Valve	quick opening	non-throttling fully open/closed	

2.3. Liquid Rocket Engine Control: Historical Background

The Apollo Lunar Module Decent Engine (LMDE) required and achieved throttling in 1968. Controlled throttling and space-vacuum restarts as well as combustion stability were key objectives during the mission. The bi-propellant engine was designed for accurate propellant injection control to be able to maintain optimum performance during the flight. Thus, a variable area pintle injector and control valves were utilized to decouple propellant flow rate and injection functions. [20]
The RL-10 by Pratt & Whitney Rocketdyne was able to operate with a different propellant combination. It was designed as a closed expander cycle. Three valves, Thrust Control Valve (TCV), Turbine Bypass Valve (TBV), and Oxidizer Combustion Valve (OCV), enabled throttling. Versions of the RL-10 engine were flown on multiple launch vehicles including Saturn I, Delta III, and different Atlas versions. [20]
The first large scale reusable rocket engine was the Space Shuttle Main Engine (SSME), able to generate up to 2091 kN thrust (vacuum). The staged combustion liquid rocket engine was powered by hydrogen and LOX, while five valves[3] were used for engine control. The actual SSME only used Fluel Preburner Oxidizer Valve (FPOV) and Oxidizer Preburner Oxidizer Valve (OPOV) as closed-loop control valves (Baseline control). For a Multivariable Control (MVC) configuration all remaining valves were considered closed-loop as well and a Oxidizer Preburner Fuel Valve (OPFV) was added.[5, 21]
Hydraulic actuator valves were used in the SSME. In case of a failure, the valves were actuated by pneumatic elements rather than by the controller. [11]
Figure 2.4 shows the flow schematic of propellants in SSME. The combustion chamber was cooled by the hydrogen, which fed the low-pressure fuel pump and bled from the high-pressure LOX-pump, which powered the low-pressure LOX-pump. The turbopumps were driven by the pre-burners, which represent the first stage of the engine. While the gas from the pre-burners is injected into the main combustion chamber, the turbopumps provide the coolant flow and transport LOX into the main combustion chamber. [5, 21]
Startup and Shutdown of the SSME were accomplished as open-loop, a scheduled control scheme based on engine simulation and testing. The closed-loop control is accomplished via Proportional Integral (PI) control. The control of MR upholds the performance and temperature of the main combustion chamber, while setpoint control of p_{CC} enables throttling. The high-pressure pump discharge pressure is regulated by the LOX flow, which passes through the LOX and fuel pre-burner. Adjusting the discharge pressure is done via Oxidizer Preburner Oxidizer Valve (OPOV) and Fluel Preburner Oxidizer Valve (FPOV) and influences p_{CC} and MR in the main combustion chamber. Direct control of the LOX and fuel preburner is impossible as Main Oxidizer Valve (MOV) and Main Fuel Valve (MFV) are fully open and Oxidizer Preburner Fuel Valve is not available on the flight model. Depending on p_{CC}, Coolant Control Valve (CCV) can be adjusted. This Baseline control of the SSME was successful, even though the MR of the pre-burners (and therefore, the temperature) could

[3] Main Oxidizer Valve (MOV), Main Fuel Valve (MFV), Coolant Control Valve (CCV), Oxidizer Preburner Oxidizer Valve (OPOV) and Fluel Preburner Oxidizer Valve (FPOV)

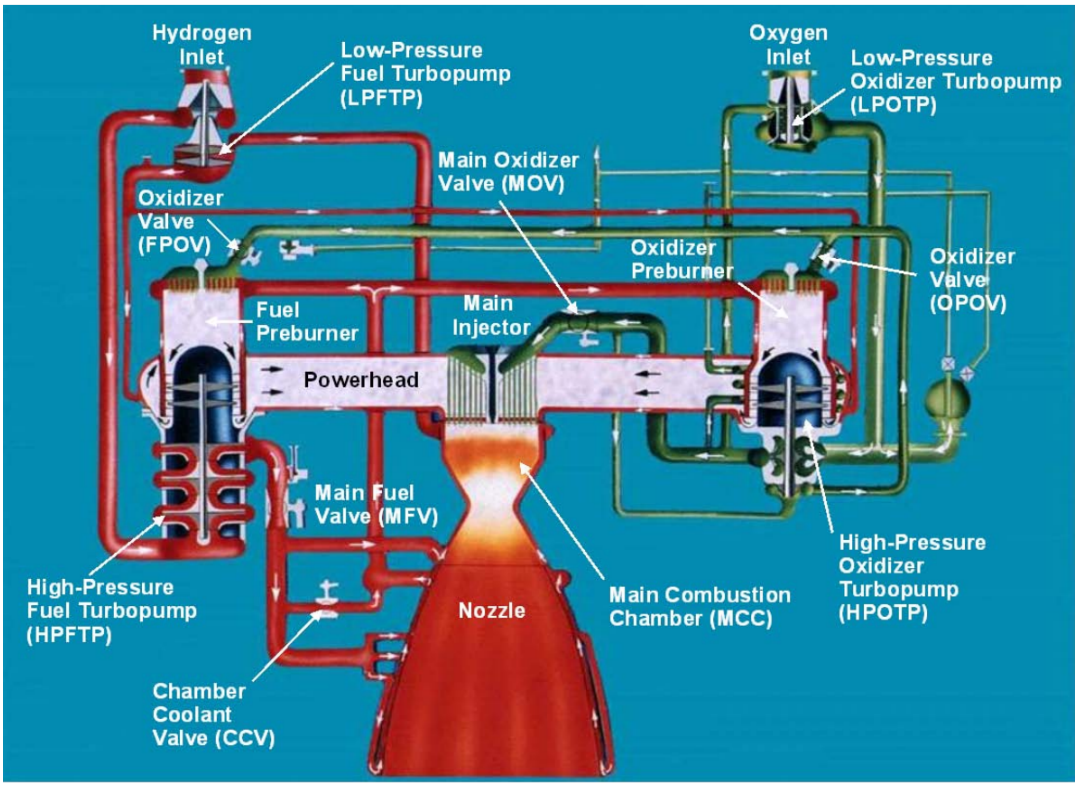


Figure 2.4.: Space Shuttle Main Engine Propellant Flow Schematic from Bradley and Hooser [21]

not directly be controlled and resulted in a shorter life-span than expected. Some issues of the SSME were the turbine blades (as the turbine temperature could not be regulated), main combustion chamber liners, propellant ducts, and bearings. The Space Shuttle Main Engine (SSME) had a thrust operating range from 50 % to 109 %, which was able to change within 1 % increment changes. The first start of the Space Shuttle took place in 1981, while testing already began in 1975. [5, 21, 22]

The cryogenic booster engine LE-X was designed for the Japanese launch vehicle H-X. It is an expander cycle and uses automatic control of thrust and mixture ratio by employing electric actuator valves and scheduled to be launched in 2021. Main Oxidizer Valve, Main Fuel Valve and the Thrust Control Valve (TCV) (also described as Coolant Control Valve (CCV)) controls thrust and MR. The gas flow that drives the turbine is regulated by TCV, consequently controlling the thrust. MR is influenced by the oxidizer pressure of the combustion chamber inlet, which can be controlled by Main Oxidizer Valve. Main Fuel Valve control is used for throttling operations to keep the turbine temperature in range and control the propellants' flow rate. Electro-mechanical actuators ensure continuous valve position control. The LE-X engine uses LOX and Liquid Hydrogen (LH_2). [14]

SpaceX's Merlin engine is a deeply throttleable engine with a gas generator cycle. It is a reuseable engine, which utilizes kerosene and LOX as propellants. The engine is employed in

the first and second stage of the Falcon 9 launch vehicle, which is also designed by SpaceX. [3]
ArianeGroup started to put a lot of effort into rocket engine control to control turbine speed, tank pressure, and other variables. Older engine generations (HM7 and Viking) used simple control systems, which contained hydro-mechanical loops. The Vulcain engine uses mono-variable control having two separated turbopumps and one gas generator. [11]
Three control valves were used to adjust the operating points: the gas generator oxygen valve and gas generator hydrogen valve controlled the thrust and the hot gas valve influenced the mixture ratio. The valves responded to pre-set mechanical stops to control thrust and MR. [23] Multi-variable control was introduced to improve the engine's performance and integrated into the Vulcain and VINCI engines in 2003. [4] The VINCI engine is a liquid propellant engine ready to fly on Ariane 6, which uses two separate turbopumps and thus two bypass valves to regulate flow rates, controlling thrust and MR. It is an expander-cycle, which uses Multivariable Control (MVC). [24]
The Prometheus engine is currently under development by ESA and will use LOX and Liquid Methane (LCH_4) as propellants. It is a traditional gas generator rocket engine, designed to be reusable and throttleable from 30 % to 110 % thrust. Autonomous thrust control and Hybrid Multi-Start (HMS) algorithms, as well as on-board computing, will utilize to improve flight performance, engine health, and post-flight maintenance. Two chamber valves control the combustion chamber MR, while two gas generator valves control the gas generator MR and thus influence the thrust. [25]

2.4. Summary

Liquid rocket engine propulsion control systems can be separated into open-loop and closed-loop control. Open-loop control offers limited control, as the engine itself does not take action upon measures engine variables [1]. Closed-loop control adjusts the system according to the sensor outputs to reach predefined setpoints. It can be used for thrust-level, MR, propellant-utilization and thrust-vector control. [1, 11]
Liquid rocket engine components are exposed to extreme thermo-mechanical loads and thus suffer from low-cycle fatigue, time-dependent damage and degradation [6]. Therefore, health monitoring to detect engine failures is crucial. The engine's performance during the flight can be adjusted to protect parts from wearing out during the flight. Closed-loop control is required to detect malfunction and take appropriate measures. [5, 11]
Multivariable Control (MVC), Intelligent Control System (ICS), and Life Extending Control (LEC) are control methods applied in reusable liquid rocket engines [11].
Valves control the propellant flow to feed engine components. By adjusting the valve settings different operational points can be reached. Valves can be driven by pneumatic, hydraulic or electric actuators. The valve characteristic describes the valve opening position in comparison to the C_v value. The different characteristics (quick-opening, linear, and equal percentage) influence the flow rate control. [17]

The Space Shuttle Main Engine was the first large-scale reusable rocket engine. It had a staged combustion cycle operating in closed-loop control and Multivariable Control (MVC), using hydraulic actuator valves. [5]
The Japanese liquid booster LE-X is an expander cycle engine using automatic control to control thrust and MR using electric actuator valves [14]. The European VINCI engine, the propulsion system of the Ariane 6, uses Multivariable Control (MVC) to improve the performance of the engine [24].

3. LUMEN

The Liquid Upper-stage deMonstrator ENgine (LUMEN) project build by the Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center) (DLR) is a component research engine, which is an expander-bleed engine powered by LOX and LNG. The engine is designed and tested at DLR Lampoldshausen. The engine is intended for test bench use, should offer the maximum amount of possibilities for regulation, and will not be flight hardware. The modular design structure gives easy access to all components, to be able to analyze each component individually. It is designed to operate at a pressure range of 35 bar to 80 bar, while holding a Mixture Ratio (MR) range of 3.0 to 3.8. The LUMEN bread-board engine is able to generate a nominal thrust of 25 kN. It is designed to represent an upper stage liquid rocket engine. [26, 27]
The main goal of the LUMEN project is to gain system level expertise, develop, and test an entire rocket engine (not only the components in itself) and to gain insight into nonlinear connections between all subsystems (such as turbopumps, cooling channel, combustion, and valves). To control the combustion chamber pressure and mixture ratio, which also defines combustion chamber temperature, injection temperature, and cooling channel pressure, a valve control sequence can be introduced.
Several control valves can be adjusted to reach the defined operating points. The optimal valve sequences for the setpoints are to be determined via Reinforcement Learning (RL) as on-board a spacecraft policy training is not realistic, due to computing, fuel, and time limitations. Instead, the policy can be trained in a simulated environment and, if accurate enough, transferred to the physical model. [28]
A LUMEN simulation model is generated with EcosimPro. With a RL algorithm described in section 4.2 and the EcosimPro simulation model, an agent can be trained to find valve sequences to meet the setpoints (chapter 5).
In this chapter LUMEN system behavior is analyzed, an EcosimPro model is generated and validated.

3.1. LUMEN Components

LUMEN is an expander-bleed engine, using two turbopumps to supply the combustion chamber. Fuel is running through a cooling channel, cooling nozzle extension, and combustion chamber before the heated fuel is used to power the turbines. A mixer is utilized to configure the injection temperature. A schematic of LUMEN can be seen in Figure 3.1.
The injector is the interface for the subsequent ignitor. It is supplied with LOX and super-critical LNG, which leads to a variation of the injection temperature. The injector is attached

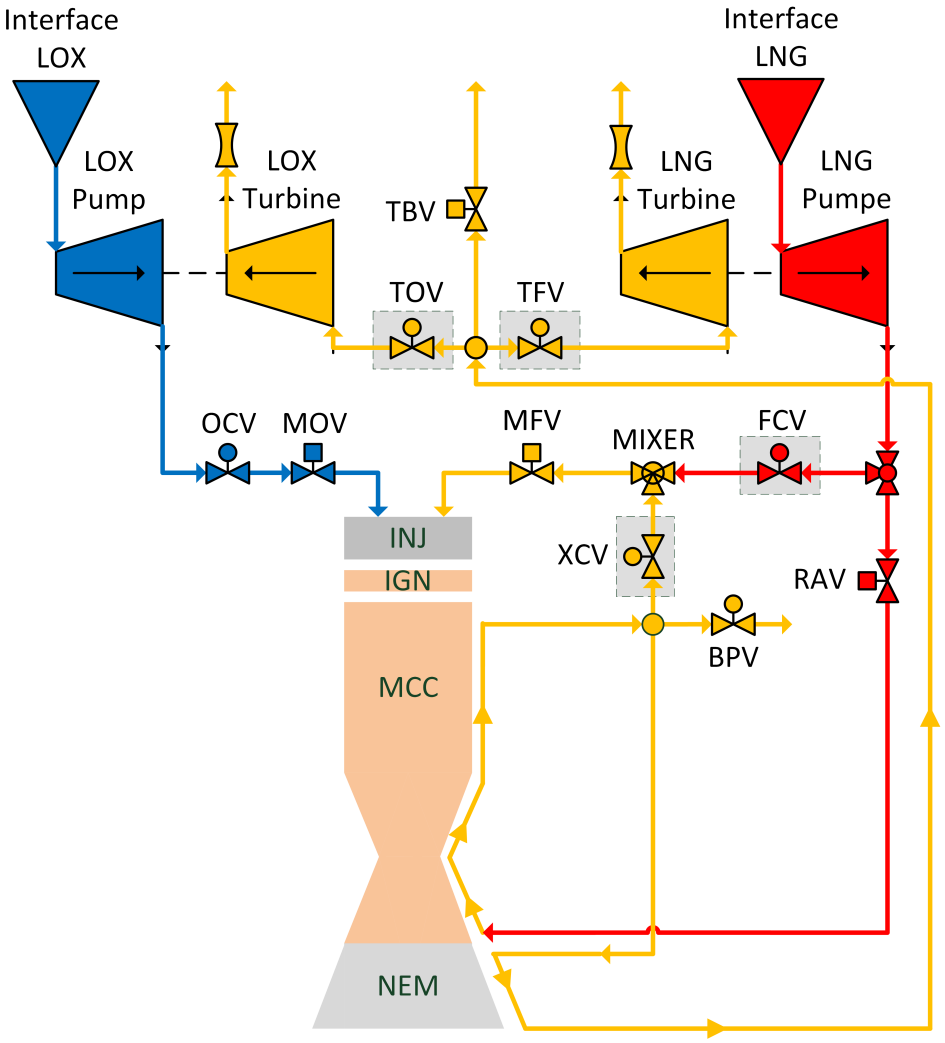


Figure 3.1.: LUMEN Schematic from Traudt, Deeken, Oschwald, and Schlechtriem [29]

to the combustion chamber, which is cooled with liquid LNG. The cooling channel runs along with the combustion chamber and the nozzle extension. Regenerative Cooling Channel non-Adjustable Valve (RAV) prefaces the cooling channel. The fuel from the cooling channel is distributed via Mixer Control Valve (XCV) to the fuel mixer and via Bypass Valve (BPV) to the bypass, where the excess propellant is dumped. Heating the fuel is vital to power the turbopumps. The combustion chamber wall material is specified as CuCrZr, which limits the maximum wall temperature to 900 K. Counter-flow cooling is established at the combustion chamber and co-flow cooling at the nozzle extension. [30]
LUMEN includes two separated turbopumps, instead of single-shaft turbopumps, which reduce weight. A detached turbopump system benefits from being able to reach the optimum efficiency for both pump systems and is easier to control. Oxidizer Combustion Valve (OCV) and Main Oxidizer Valve (MOV) are located downstream the pump on the oxidizer side.

The LNG flow after the turbopump system is split up into Fuel Control Valve (FCV) and Regenerative Cooling Channel non-Adjustable Valve (RAV). The turbines are driven by the heated fuel from the regenerative cooling system. Turbine Oxidizer Valve (TOV) and Turbine Fuel Valve (TFV) regulate the propellant flow into the turbines, which drive the pumps. The propellant mixer enables to remix a part of the heated fuel from the regenerative cooling system. The injector is fed with gaseous or super-critical fuel, which is provided by mixing liquid propellant from Fuel Control Valve (FCV) and gaseous propellant from XCV. The Main Fuel Valve (MFV) connects the propellant mixer outlet and injector.
The various electric actuated control valves incorporated into the system offer high control flexibility. At the same time, an increasing number of valves leads to higher control efforts. During the development of the LUMEN demonstrator, some valves are to be replaced by throttle components. [31, 32]

3.2. Operating Points

A large throttling range is one of the design goals of LUMEN. The nominal operation point of LUMEN is at 60 bar combustion chamber pressure while holding $MR = 3.4$. The engines boundary load points are 35 bar and 80 bar. MR has a range of 3.0 to 3.8. The engines throttling range is 58 % to 133 %. [33]
The operating points of the LUMEN demonstrator can be seen in Table 3.1.

Table 3.1.: Operating Points of the LUMEN Demonstrator from Hardi, Martin, Son, et al. [26]

	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9
Combustion Chamber Pressure [bar]	60	80	35	60	80	35	60	80	35
Mixture Ratio	3.4	3.4	3.4	3.0	3.0	3.0	3.8	3.8	3.8

Table 3.2.: Constraints of the LUMEN Demonstrator [26, 32]

	Minimum Value	Maximum Value
LOX Turbine Speed [rpm]	/	30 000
LNG Turbine Speed [rpm]	/	50 000
Turbine Inlet Pressure [bar]	30	/
LNG Pump Outlet Pressure [bar]	/	150
Cooling Channel Pressure [bar]	46	/
LNG Injection Temperature [K]	190	/
Combustion Chamber Wall Temperature [K]	/	900

During engine operation, constraints, seen in Table 3.2, cannot be violated. The rotational speed of the turbines is set due to mechanical limits and with enough distance to its natural

frequency by Traudt, Mason, Deeken, et al. [27].
The minimum LNG injection temperature is set to 190 K to meet the gaseous or super-critical fuel inlet condition. [33]
LNG consists of methane (CH_4) and might contain ethane (C_2H_6). The critical point for methane is reached at 46 bar. To ensure that the LNG remains liquid the pressure cannot fall below this critical point. [34]
An appropriate cooling channel mass flow rate needs to be chosen to not exceed the maximum wall temperature.

3.3. EcosimPro/ESPSS Model

EcosimPro is a simulation tool, which offers modeling and simulation for various complex dynamic systems. A set of libraries is employed by EcosimPro, which contains different types of components and can be included in the model. [35]
European Space Propulsion System Simulation (ESPSS) contains multiple libraries including various propulsion system components and is used to adjust the already existing EcosimPro model from Traudt, Waxenegger-Wilfing, Santos Hahn, et al. [36]. The Ecosim model of the demonstrator can be seen on page 20. It shows the connection between the different components of the engine. The pipes are implemented to represent the time delays and dynamics, and therefore performance losses between components. The heat flow multiplier is implemented to adjust the heat pick-up in the curved cooling channel compared to straight cooling channels. The three most common types of valve flow characteristics are supported by EcosimPro. In the EcosimPro Model RCV is changed to RAV as the valve is not adjustable. MOV and MFV are only used for start-up and shut-down of the engine, hence the valve flow characteristics are set to quick opening. TOV and TFV regulate the turbine flow and have a linear flow characteristic. As all other valves are used for precise regulation, equal percentage is used as the valve flow characteristic.
The flow resistance ζ for all valves can be seen in equation 2.5, using the discharge coefficient c_D. [18] The K_v value is calculated according to equation 3.2 [32]. A_1 and A_2 represent the flow areas of the connected pipes and can also be described as the inlets and outlets of the valve.

$$\zeta = \frac{1}{0.3308^2} = 9.138375475 \tag{3.1}$$

$$K_v = \sqrt{\frac{d_0^4}{635.439 * \zeta}} \tag{3.2}$$

Another important value regarding the valves is the valve opening time constant τ. τ the time constant of the first order transfer function. The delay time for the commanded valve position is calculated according to equation 3.3. It models the actuator, which controls the valve. The valve position pos as well as the commanded valve position pos_{com} are non-dimensional values, in a range from 0.0 to 1.0. τ is set to 0.2 s after the start-up as it correlates with the

true values of the used valves. [18, 37]
The pressure loss above the orifice after BPV can be calculated according to equation 2.6.

$$\frac{pos}{dt} = \frac{(pos_{com} - pos)}{\tau} \tag{3.3}$$

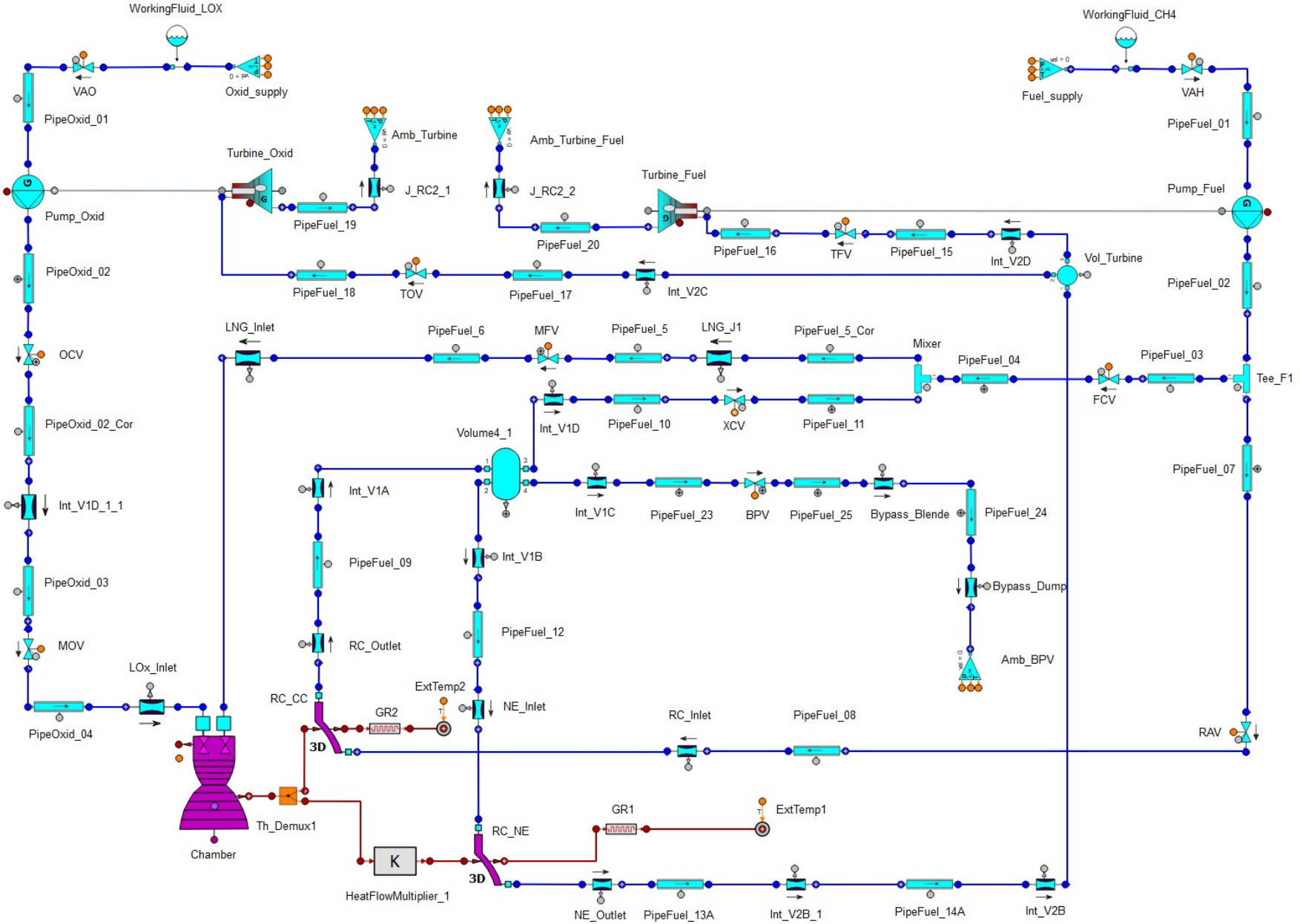
In EcosimPro the generic turbine and pump from European Space Propulsion System Simulation (ESPSS) are used, in which the design point is set and off-design characteristics are adjusted automatically. The turbine's power is calculated according to equation 3.4, depending on the inlet mass flow rate and inlet and outlet enthalpy. The operating conditions define the geometrical design. Characteristic radius and inter-blade flow area scale the performance maps according to the design conditions, which include e.g. efficiency and nominal characteristic speed. [18]

$$P = \eta \dot{m}_{in}(H_{out} - H_{in}) \tag{3.4}$$

The dimensions and characteristics of the pumps are customized. The pump outlet pressure is calculated according to equation 3.5, taking inlet pressure and total dynamic head into account. The optimal axial speed is set to dynamic and hence dynamically calculated. [18]

$$p_{out} = 9.806\rho_{in}TDH + p_{in} \tag{3.5}$$

The *CombustChamberNozzle* component includes an injector, combustion chamber, and nozzle. After the injector, combustion chamber, and nozzle dimensions are set, the number of subsonic and supersonic nodes for the fluid and thermal calculation has to be defined. The number of nodes influences the depth of the fluid and thermal simulation. Increasing the node count too much can lead to an unstable and diverging simulation as well as prolonging the calculations. A non-adiabatic approximation is used to simulate the flow in the nozzle. Within the combustion chamber, no convection of the liquid propellants is calculated, which means that vapor is formed after ignition within a time delay. To simplify the mathematical model (transient conservation equations) of the combustion and expansion process, a 1D quasi-steady implicit method is applied, which includes non-isentropic effects under equilibrium or frozen conditions. First, the throat section is calculated with the exit conditions of the combustion. The calculations within the nozzle component (supersonic section) are separated into two steps. First, enthalpy and entropy are calculated using Bartz correlations and then the expansion process can be calculated. In the last step, thrust and I_{sp} are determined. [18, 38]
The combustion chamber and nozzle extension are connected to a cooling channel, which simulates the walls of the chamber and nozzle. The exit conditions of the combustion chamber are transmitted via the nozzle port. The combustion chamber cooling channel uses supersonic and subsonic nodes, while the nozzle extension cooling channel is only connected to the supersonic nodes.[18]


WorkingFluid_LOX
VAO
Oxid_supply
PipeOxid_01
Amb_Turbine
Turbine_Oxid
J_RC2_1
Amb_Turbine_Fuel
J_RC2_2
Turbine_Fuel
Pump_Oxid
PipeFuel_19
PipeFuel_20
PipeFuel_16
TFV
PipeFuel_15
Int_V2D
Vol_Turbine
PipeOxid_02
PipeFuel_18
TOV
PipeFuel_17
Int_V2C
LNG_Inlet
PipeFuel_6
MFV
PipeFuel_5
LNG_J1
PipeFuel_5_Cor
Mixer
OCV
Int_V1D
PipeFuel_10
XCV
PipeFuel_11
PipeFuel_04
FCV
PipeFuel_03
Tee_F1
PipeOxid_02_Cor
Volume4_1
Int_V1A
Int_V1C
PipeFuel_23
BPV
PipeFuel_25
Bypass_Blende
PipeFuel_24
Int_V1D_1_1
PipeFuel_09
Int_V1B
Bypass_Dump
PipeOxid_03
PipeFuel_12
Amb_BPV
MOV
RC_Outlet
LOx_Inlet
ExtTemp2
RC_CC
GR2
NE_Inlet
RC_Inlet
PipeFuel_08
RAV
PipeOxid_04
3D
Th_Demux1
Chamber
GR1
ExtTemp1
RC_NE
K
HeatFlowMultiplier_1
3D
NE_Outlet
PipeFuel_13A
Int_V2B_1
PipeFuel_14A
Int_V2B
WorkingFluid_CH4
Fuel_supply
VAH
PipeFuel_01
Pump_Fuel
PipeFuel_02
PipeFuel_07

A thermal demultiplexer splits the single thermal port from the combustion chamber and nozzle component into two different thermal ports to be able to connect separate cooling channels. LUMEN uses a cooling channel for the combustion chamber and a separate cooling channel for the nozzle extension. [18]
The heat flow multiplier multiplies the inlet heat with a factor k, which produces the outlet heat to model the curved cooling channel geometry. Inlet and outlet temperatures are identical. [39]

$$q_{outlet} = kq_{inlet} \tag{3.6}$$

A volume is used as a four-way junction. The fluid volume is set to $0.0003\,\mathrm{m}^3$ to avoid cavities. EcosimPro requires an engine startup at each model execution. The startup is considered complete in this case, after the system reaches thermodynamic equilibrium. The startup is saved and can be called up as the initial state for the calculations in this thesis. Engine startup is not part of this thesis.

3.3.1. LUMEN System Analysis

The impact of valve adjustment is analyzed in this section to understand and visualize the system dynamics. Starting at fixed initial state, one valve at a time is opened/closed[4] for 10 s and brought back to its initial position within 10 s. The position change for each valve is 0.2. Table 3.4 shows an overview of the impact of valve adjustment on the system variables. Visualization of valve adjustment impacts and the system response can be seen in Figure 3.2 to 3.7.
For better understanding of this section the valve acronyms and its meanings are displayed in table 3.3.

Table 3.3.: Overview Valve Acronyms

Acronym	Meaning	Acronym	Meaning
BPV	Bypass Valve	RAV	Regenerative Cooling Channel non-Adjustable Valve
FCV	Fuel Control Valve		
MFV	Main Fuel Valve	TFV	Turbine Fuel Valve
MOV	Main Oxidizer Valve	TOV	Turbine Oxidizer Valve
OCV	Oxidizer Combustion Valve	XCV	Mixer Control Valve

[4]Only OCV is closed due to its initial position of 1.0

Table 3.4.: Impact of Valve Opening Adjustments

↑: increasing; ↓: decreasing; -: constant Variable

Variable	XCV ↑	TOV ↑	TFV ↑	FCV ↑	OCV ↑	BPV ↑
Combustion Chamber Pressure	↓	↑	↑	↑	↑	↓
Mixture Ratio	↓	↑	↓	↓	↑	↑
LNG Injection Temperature	↑	↓	↓	↓	↑	↓
Cooling Channel Mass Flow Rate	↑	↓	↑	↓	↓	-
Cooling Channel Outlet Pressure	↓	↓	↑	↓	↑	↓
Cooling Channel Outlet Temperature	↓	↓	↓	↑	↑	↓
MOV Mass Flow Rate	↓	↑	↓	↓	↑	↓
MFV Mass Flow Rate	↑	↓	↑	↑	-	↓
LNG Turbine Speed	↓	↓	↑	↓	↑	↓
LOX Turbine Speed	↓	↑	↓	↓	↓	↓
LNG Pump Outlet Pressure	↓	↓	↑	↓	↑	↓
LOX Pump Outlet Pressure	↓	↑	↓	↓	↓	↓

TFV: Opening TFV increases the mass flow to the LNG turbopump and thus increases the LNG pump power. As the amount of fluid transported by the LNG pump increases, mass flow through RAV, the cooling channel, increases (Figure 3.2). The more fluid is pumped through the cooling channel, the lower the cooling channel temperature. The cooling channel pressure decreases as mass flow increases through RAV and BPV is being closed, mass flow through XCV increases as well. FCV mass flow stays constant. As a result, the mass flow downstream of the mixer increases. As the cooling channel temperature drops, the LNG injection temperature drops as well. A higher mass flow through TFV causes a lower mass flow through TOV, which decreases the LOX pump power and hence mass flow through MOV. MR decreases as a result of a significantly lower mass flow rate through MOV and a higher mass flow through MFV. As the injection pressure decreases the combustion chamber pressure decreases as well.

XCV: Opening XCV causes the MR to drop and the combustion chamber pressure to slightly decrease (Figure 3.3). The pressure in the cooling channels drops, which causes the mass flow rate towards the turbines to decrease and subsequently the rotational speed of the turbines and thus the outlet pressure of the pumps to drop. The mass flow rate through FCV decreases, which leads to a higher amount of heated fuel from the cooling channel running through the mixer. The injection temperature of the fuel rises. Due to the decreasing pump speed, the mass flow rate through OCV decreases. As the mass flow rate through MFV increases, the mixture ratio drops.

TOV: Opening TOV leads to an increase of MR and combustion chamber pressure (Figure 3.4). The inlet pressure into the LOX turbine and therefore the rotational speed increases. As the turbines are fed through the same supply, the inlet pressure of the LNG turbine decreases, which also reduces the rotational speed. The pumps react accordingly. The pressure loss at the injection thus increases for the oxidizer and decreases for the LNG. As

LNG pump speed decreases the pressure in the cooling channels drops. The LNG turbine is fed by the nozzle extension cooling channel flow, which's pressure decreases. The LNG temperature after the nozzle extension decreases.

BPV: Opening BPV results in combustion chamber pressure decrease and a slight increase in MR (Figure 3.5). The pressure in the cooling channel drops, resulting in lower rotational speed of both turbines, and thus lower outlet pump pressure on oxidizer and fuel side. As injection pressure of both propellants decrease, the combustion chamber pressure also drops.

OCV: Combustion chamber pressure and MR increase when OCV is opened (Figure 3.6). Cooling channel pressure increases resulting in an increase of MFR through TOV and TFV, leading to higher outlet pressure in both pumps.

FCV: The combustion chamber pressure increases and MR decreases as FCV is opened (Figure 3.7). The cooling channel pressure decreases, which results in lower rotational speed in both turbines and a lower pump outlet pressure on both sides. The cooling channel mass flow rate drops, causing the fuel to heat up and the turbine inlet temperature to rise. The injection temperature of the fuel decreases, as a result of FCV letting pass more fluid into the mixer.

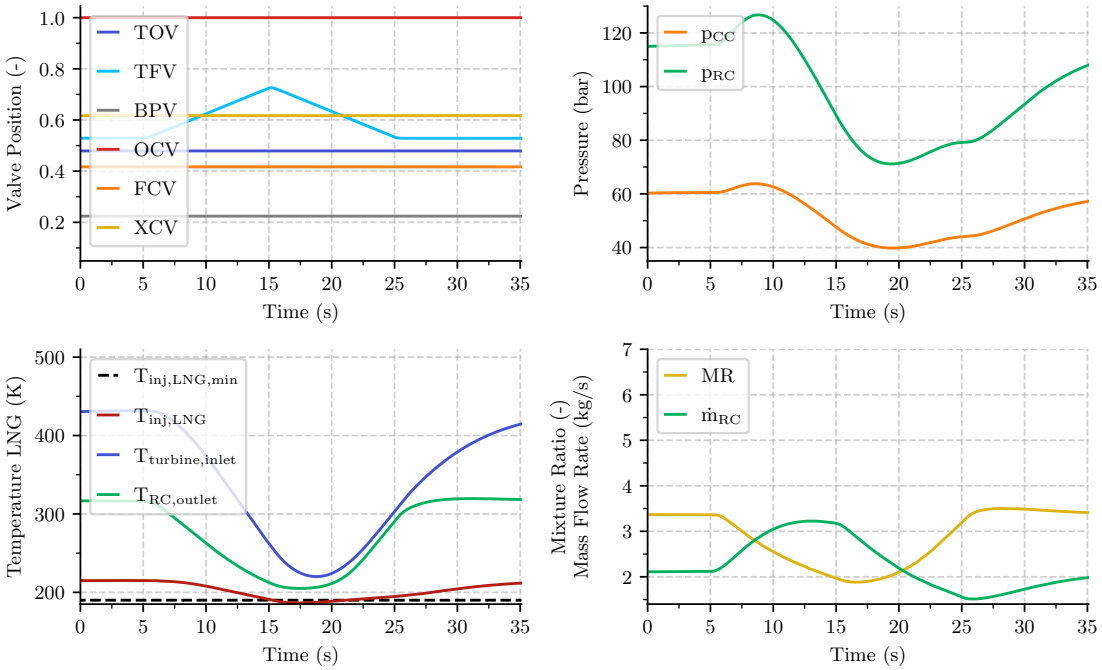


Figure 3.2.: LUMEN Turbine Fuel Valve Adjustment

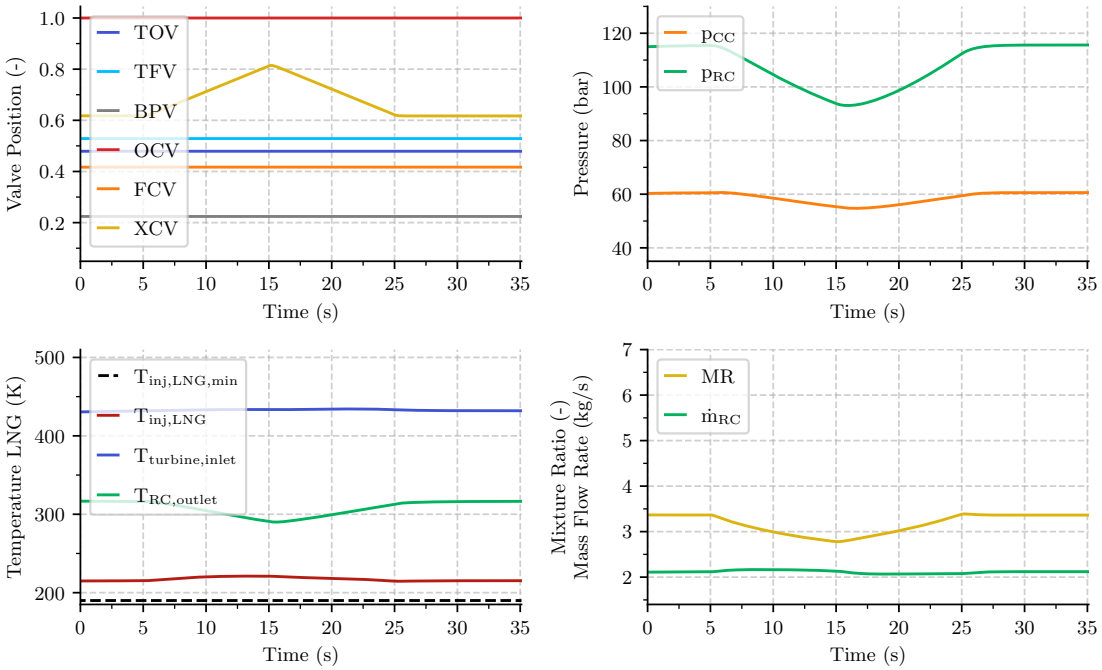


Figure 3.3.: LUMEN Mixer Control Valve Adjustment

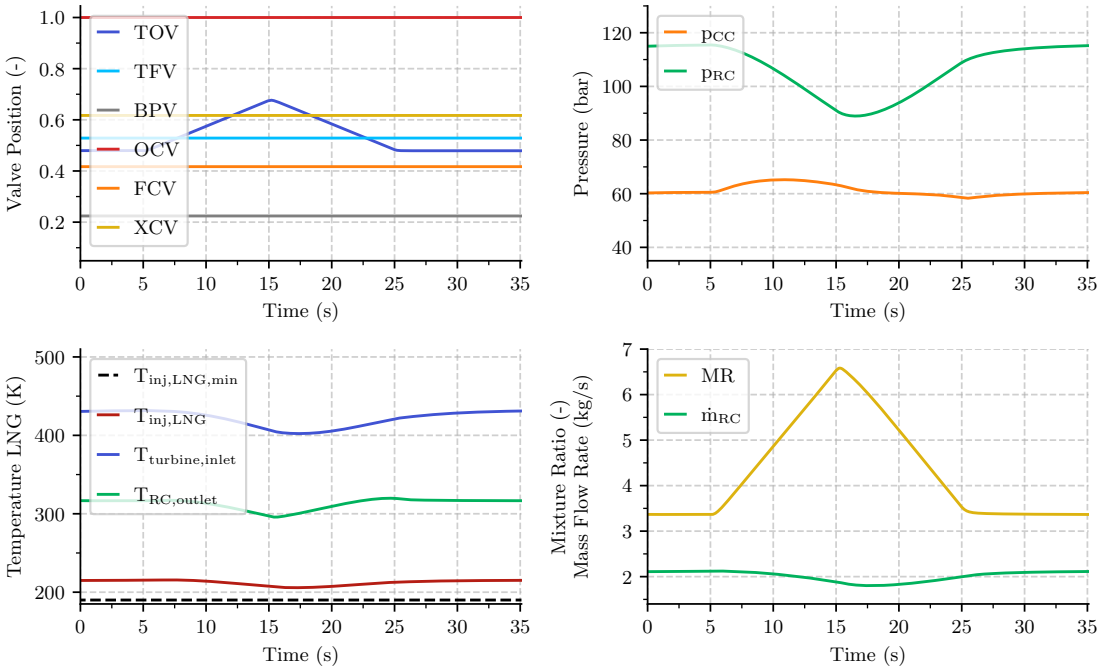


Figure 3.4.: LUMEN Turbine Oxidizer Valve Adjustment

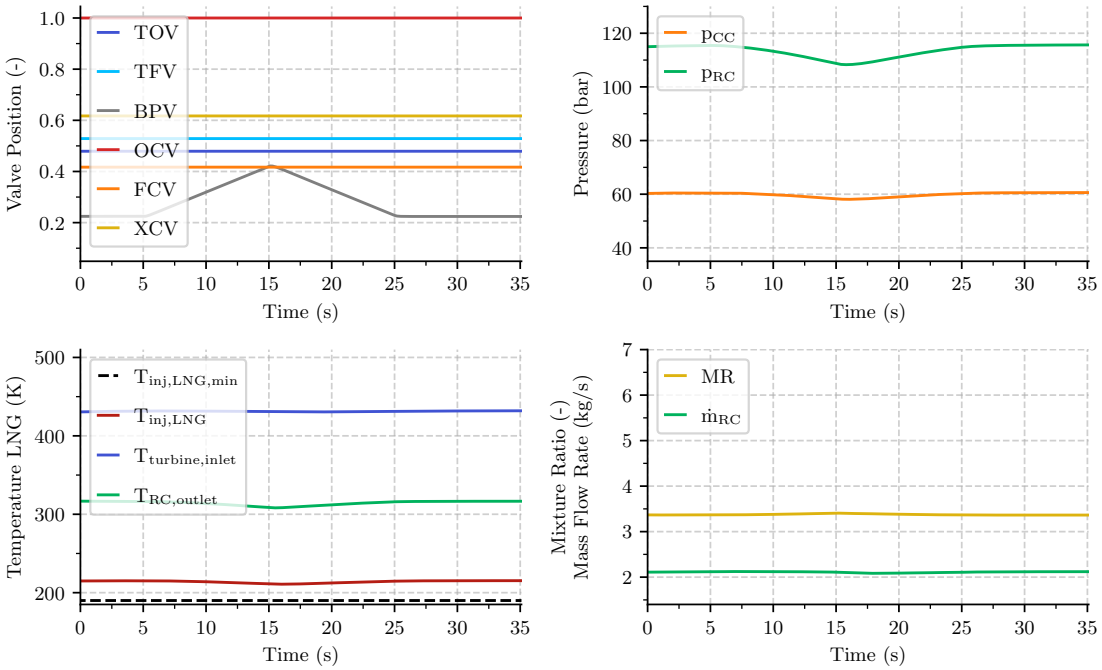


Figure 3.5.: LUMEN Bypass Valve Adjustment

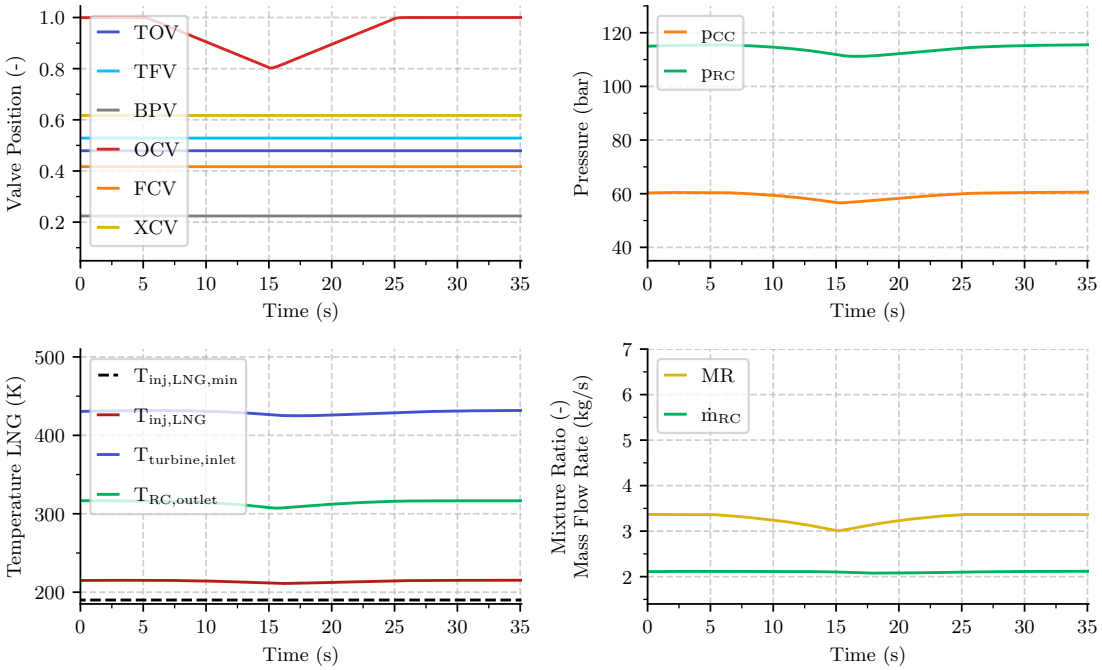


Figure 3.6.: LUMEN Oxidizer Combustion Valve Adjustment

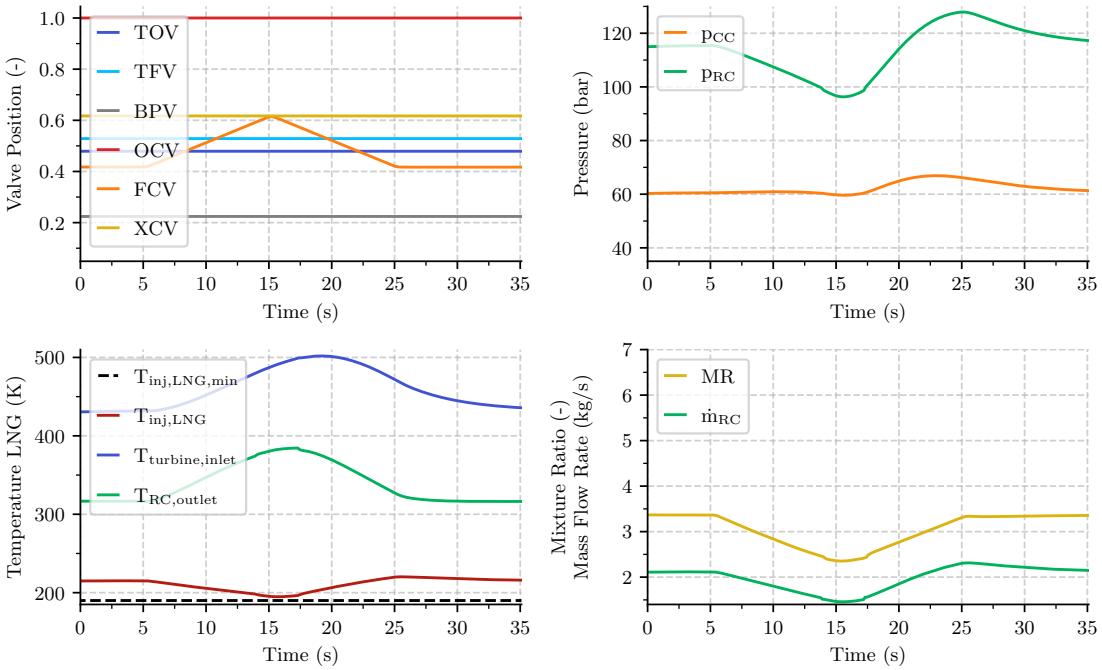


Figure 3.7.: LUMEN Fuel Control Valve Adjustment

3.3.2. LUMEN System Validation

The manipulation of one valve leads to changing variables throughout the engine system, as explained in section 3.3.1. Due to the complex system, it is difficult to reach operation points by changing valve settings by hand. To validate the EcosimPro, exact operation points may not be reached by manually adapting valve positions, as the system is too complex and changing a single setting causes a variance in most output variables. Table 3.5 shows the set points in comparison to manually reached operation point 1 using EcosimPro.
Manually adjusting the valves to reach operation point 1 was accomplished with a derivation of less than 15 %. MR was achieved with 1.5 % derivation, while combustion chamber pressure derivation is below 6 %. The main focus was to achieve accuracy in combustion chamber pressure and MR, thus other variables, like the cooling channel outlet pressure, deviate slightly more. Turbine rotational speed is within the limits.
As the EcosimPro Simulation for the cooling channel wall temperature is based on simple correlations it might not be accurate enough. To avoid this problem and implement a more accurate cooling channel wall temperature prediction into the model, a neural network could be employed to compute the temperature in future applications [31].

Table 3.5.: EcoSimPro Validation: Operation Point 1 (Set Point Values from Deeken, Waxenegger-Wilfing, and Santos Hahn [32])

Variable	Mixture Ratio [-]	Combustion Chamber Pressure [bar]	Cooling Channel MFR [kg/s]	Injection Temperatuer LNG [K]	Fuel Turbine Rotational Speed [rpm]	Oxidizer Turbine Rotational Speed [rpm]
Setpoint	3.4	60	1.7	192.6	/	/
EcosimPro	3.35	63.42	1.71	195.14	34967	24154

Variable	Injection Temperature LOX [K]	MOV MFR [kg/s]	MFV MFR [kg/s]	Cooling Channel Wall Temperature [K]	Cooling Channel Outlet Pressure [bar]	Fuel Turbine MFR [kg/s]	Oxidizer Turbine MFR [kg/s]
Setpoint	97.0	5.8	1.7	900	81.5	0.5	0.5
EcosimPro	100.46	6.06	1.81	779.47	88.77	0.43	0.51

3.4. Summary

The Liquid Upper-stage deMonstrator ENgine (LUMEN) is an expander-bleed engine, designed as a test bench research engine by DLR Lampoldshausen. It is powered by LNG and LOX and can be operated at different operational points, reaching combustion chamber pressures of 35 bar to 80 bar and Mixture Ratio (MR) from 3.0 to 3.8 are achievable. [26, 27]
The LNG pump of LUMEN feeds the regenerative cooling system of the combustion chamber and nozzle extension before the heated fuel reaches the turbopump systems (figure 3.1). High pressure pumps transport LOX and LNG into the combustion chamber. LUMEN has six adjustable valves, which can be manipulated to reach various set points. The Bypass Valve (BPV) is a further bleed component of the engine and enables an exhaust flow of LNG to enable low combustion chamber pressure operational points.
An EcosimPro model from Traudt, Waxenegger-Wilfing, Santos Hahn, et al. [36] was altered to include all LUMEN components. Validation of the model was conducted by manually adjusting valve positions. Operation point 1 was reached with less than 15 % derivation of the steady state model from Deeken, Waxenegger-Wilfing, and Santos Hahn [32], which is within acceptable limits.

4. Reinforcement Learning

Reinforcement Learning (RL) is part of artificial intelligence and machine learning. In RL an agent learns how to act in a given environment to maximize a defined reward. The agent is not told which action to take and must discover the actions which result in the highest reward. The implemented algorithm on which the agent acts upon, studies the behavior and consequences in an environment to optimize the behavior and maximize the reward. Reinforcement learning is often applied in game theory, computer games, and robotics. The process of reinforcement learning can be described as a Markov Decision Process which is illustrated in Figure 4.1. [8]
The following information in this chapter are taken from Sutton and Barto [8] and Li [40] unless labeled differently.

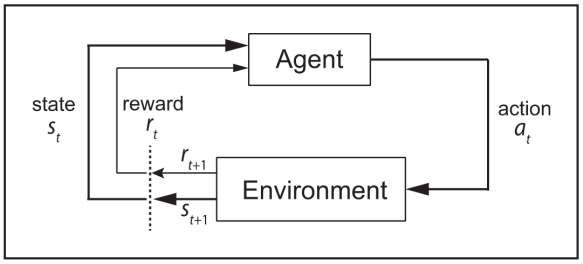


Figure 4.1.: Markov Decision Process from Sutton and Barto [8]

4.1. Fundamentals of Reinforcement Learning

An **agent** in reinforcement learning is the decision maker, it is interacting with the given **environment** to achieve a defined goal. Every decision which is made can be considered an **action** a_t. The environment's observation at each time step is referred to as the **state** s_t. Given the state of the environment, the agent can select its action, which changes the environments state to a new state. As a consequence of its action, the agent is given a **reward** r_t (equation4.1).

$$r_t = R(s_t, a_t, s_{t+1}) \tag{4.1}$$

When a sequence of states and actions in the reinforcement learning environment is described, it can be characterized as a **trajectory**, which might also be called rollout or episode (equation 4.2 τ, Figure 4.2).

$$\tau = (s_0, a_0, s_1, a_1, ...) \tag{4.2}$$

Observing the state, taking an action and receiving a reward is a repetitive process. The

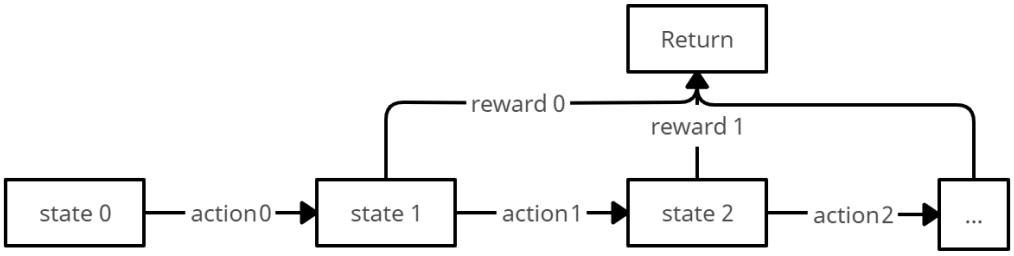


Figure 4.2.: Reinforcement Learning Trajectory

return R_t is the sum of the expected rewards during the upcoming time steps (equation 4.3). [8, 41, 42]

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_{t+n} = \sum_{t=0}^{T} r_t = R(\tau) \tag{4.3}$$

The agents objective is to maximize the return. The discount factor $0 \leq \gamma \leq 1$ insures that the return remains finite for never-ending episodes (equation 4.4).

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{t=0}^{\infty} \gamma^t r_t = R(\tau) \tag{4.4}$$

In the MDP a good policy π needs to be found. The policy function describes the action, which the agent will take according to the state (equation 4.5). The policy alone is responsible for the agents' behavior. If the policy outputs are computational functions, which depend on a set of parameters, θ is used to represent the parameters of the policy. Those parameters could for example represent the weights and biases of a neural network.

$$a_t = \pi(s_t) = \pi_\theta(s_t) \tag{4.5}$$

The optimal policy is the policy, which maximizes the total reward over an episode. The reward is only given at each timestep. Another measure is necessary to evaluate the long-term performance of the policy. One possibility is the value function, which indicates if the action was good in the long run, while the reward only evaluates the last action. The value of the state is the sum of all rewards which the agent can expect to achieve in the future. The expected return, given a state and following the selected policy, is called **value function** V and can be defined by equation 4.6. The value function indicates, how expedient the current state is. When the state and the action are given under the selected policy, the expected return is defined as the **action-value function** (equation 4.7).

$$V^\pi(s) = \mathop{\mathbb{E}}_{\tau \sim \pi}[R(\tau)|s_o = s] \tag{4.6}$$

$$Q^{\pi}(s,a) = \max_{\pi} \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_o = s, a_o = a] \tag{4.7}$$

The optimal value or action-value function can be detected when using the optimal policy (equation 4.8 and 4.9).

$$V^*(s) = V^{\pi*}(s) \tag{4.8}$$

$$Q^*(s,a) = Q^{\pi*}(s,a) \tag{4.9}$$

The optimal action-value function is also known as the optimal **Q-function** Q. While the state is given and the agent acts upon the optimal policy, an action is taken which maximizes the expected return, which optimizes the Q-function. The optimal action can therefore be described as stated in equation 4.10.

$$a^*(s) = \arg\max_{a} Q^*(s,a) \tag{4.10}$$

In addition equation 4.11 shows the **Bellman optimality equation**, which expresses the relation of optimum value function and optimum action-value function:

$$V^{\pi*}(s) = \max_{a} Q^{\pi*}(s,a) \tag{4.11}$$

The value functions must fulfill a self-consistency, which is given by the **Bellman equations**. The value of the state can be decomposed into immediate reward $r(s,a)$ plus the value of the successor state $V(s')$ with the discount factor γ. The next state is retrieved from the transition rules of the environment, while the subsequent action is chosen according to the policy. The Bellman equations for the value functions can be seen in equation 4.12 and 4.13.

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{s'\sim P}[r(s,a) + \mathop{\mathbb{E}}_{a\sim\pi} \gamma V^{\pi}(s')] \tag{4.12}$$

$$Q^{\pi}(s,a) = \mathop{\mathbb{E}}_{s'\sim P}[r(s,a) + \mathop{\mathbb{E}}_{a\sim\pi} \gamma Q^{\pi}(s',a')] \tag{4.13}$$

The Bellman optimality equations for the value functions can be seen in equation 4.14 and 4.15

$$V^*(s) = \max_{a} \mathop{\mathbb{E}}_{s'\sim P}[r(s,a) + \gamma V^*(s')] \tag{4.14}$$

$$Q^*(s,a) = \mathop{\mathbb{E}}_{s'\sim P}[r(s,a) + \gamma \max_{a'} Q^*(s',a')] \tag{4.15}$$

The **model** of the environment represents the behavior of the environment on which the agent acts upon. Given a state and action, the model predicts the next imminent state. The **observation** is the description of a state. If the complete state of the environment can be observed by the agent, the environment is fully observable. Otherwise, it is only partially observable.

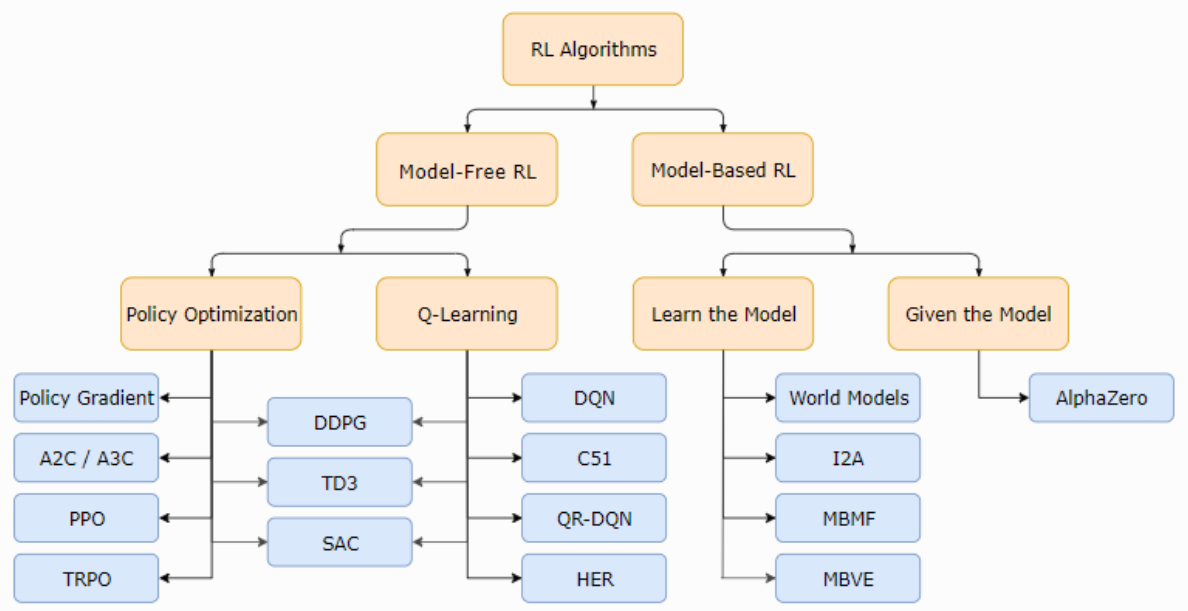


Figure 4.3.: Taxonomy of Algorithms from OpenAI [43]

4.2. Reinforcement Learning Algorithms

Reinforcement Learning algorithms can be roughly categorised in model-free and model-based RL, according to Figure 4.3.

4.2.1. Model-based and Model-free Reinforcement Learning

A model of the environment can be defined as anything the agent could use to predict the environment's response as a result of its actions. The model produces the next state and the next reward when a state-action pair is given. Sample models will return a possible transition, while distribution models produce all possible transitions and weigh them by their probable occurrence. Sample and distribution models are used to simulate the environment and generate a simulated experience. [8, 40]
Model-based RL relies on a model of the environment. The model can be predefined or explicitly learned by the algorithm. The model is a function, which can predict the state transitions and rewards. If the agent has access to a model, the agent can look at future events. Model-based RL methods can be roughly subdivided into learning the model and using a given model.
In model-free RL the agent is not dependent on the model during the learning process. It has to learn the model by interacting with the environment. One downside to this approach is that the agent might exploit model characteristics that only exist in the simulation model, but not in the real world model. The agent will then be able to perform well towards the learned model but will perform bothersome in a real environment. Model-free RL methods are computationally more expensive due to their sample complexity. In this thesis a model-free

approach was implemented, hence the following algorithm categorization will concentrate on those. Model-free RL can be divided into policy optimization and Q-learning, depending on the approach. The different categories of algorithms can be seen in Figure 4.3. [8, 40]

4.2.2. Policy Optimization

Policy optimization methods optimize the parameters θ of a policy $\pi_\theta(a|s)$. The parameters can be either manipulated indirectly by maximizing the local approximations or directly by gradient ascent of the performance objective of the expected return. Policy optimization can be performed on-policy or off-policy. On-policy optimization means, that the agent chooses actions according to the latest version of the policy and the policy is updated by only using the data collected. Policy optimization can directly optimize the agents' performance, which makes this method stable and reliable. The policy is optimized by gradient ascent[5] (equation 4.16) or gradient descent.

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\pi_\theta) \tag{4.16}$$

Policy gradient methods optimize the policy through the gradient of policy performance $\nabla J(\pi_\theta)$. Some methods additionally learn an approximate value-function, those methods are called actor-critic methods (see 4.2.5). On-policy learning is considered inefficient regarding sample collection since new samples need to be collected for each gradient step. In contrast, off-policy methods can reuse past experiences. [8, 40]
The difference between on- and off-policy RL is that on-policy methods pursue to improve the given policy (the current policy, which is used to make decisions), while off-policy algorithms improve or evaluate policies different from the one used to create data. [8]

4.2.3. Q-Learning

Q-learning is a value-based reinforcement learning algorithm, which is used to find the optimal action-selection policy using a Q-function. Q-learning methods are usually performed off-policy, which means that any point during the training can be used for each update. A new approximator $Q_\theta(s, a)$ is learned to find the optimal action-value function. Q-learning estimates the value of $Q^*(s, a)$ by using Temporal Differences (TD). The agent chooses the action with the maximum Q-value, with the highest the expected return. Equation 4.17 shows the agent chooses the action with the maximum Q-value.

$$a(s) = \arg \max_a Q_\theta(s, a) \tag{4.17}$$

Q-learning methods tend to be more sample efficient than policy optimization methods, as they can reuse data more efficiently. The agent observes the current state s_t, performs a selected action a_t, observes the subsequent state s_{t+1}, receives an reward r_t, and updates the

[5] Gradient Ascent means maximizing the loss function instead of minimizing it.

Q-function at each step t. As $t \to \infty$ the Q-function reaches the optimal Q-function (equation 4.18).

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \underset{a}{max} Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{4.18}$$

The Q-function will update until a final state is reached, this state can also be considered as terminal. [44] For better understanding a short comprehensive example is made: four cities are connected by pathways according to Figure 4.4. The goal is to reach city 4, no matter from

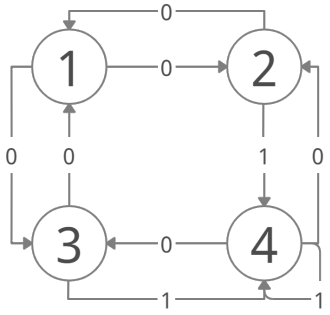


Figure 4.4.: Q-learning Example: four Cities and connecting Paths

which initial city. Reaching city 4 results in a reward of 1 and every other path results in a reward of 0. If there is no connecting path between the cities, the reward is -1. The rewards for each transition between the cities can be written in a reward matrix R:

$$R = \left[\begin{array}{cccc|c} -1 & 0 & 0 & -1 & 1 \\ 0 & -1 & -1 & 1 & 2 \\ 0 & -1 & -1 & 1 & 3 \\ -1 & 0 & 0 & 1 & 4 \\ \hline 1 & 2 & 3 & 4 & \\ \multicolumn{4}{c|}{actions} & \end{array}\right] \quad states \qquad Q = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 4 \\ \hline 1 & 2 & 3 & 4 & \\ \multicolumn{4}{c|}{actions} & \end{array}\right] \quad states$$

The learning rate and the discount factor are set to $\alpha = 0.7$ and $\gamma = 0.9$ for this example. Starting in city 1 a random action $a = 3$ is chosen, which means, that city 3 is approached. The Q-value ($Q(1,3)$) for this action can be calculated according to equation 4.18.
Being in city 3 leaves two possible actions for the next step: going back to city 1 ($Q(3,1)$) or approaching city 4 ($Q(3,4)$). The action, which returns the best reward is chosen. The Q-value for city 3 can be calculated (equation 4.20). As the Q-matrix was initialized with 0, all possible Q-values are 0 at this point.

$$Q^{new}(1,3) = Q(1,3) + 0.7[0 + 0.9 \underset{a}{max}[Q(3,1), Q(3,4)] - Q(1,3)] \tag{4.19}$$

$$Q^{new}(1,3) = 0 + 0.7[0 + 0.9 max[0,0] - 0] = 0.63 \tag{4.20}$$

In the next step city 4 is approached ($Q(3,4)$). The next possible states would be city 2, city 3 or city 4. The new Q-value for $Q(3,4)$ is calculated in equation 4.22.

$$Q^{new}(3,4) = Q(3,4) + 0.7[1 + 0.9\underset{a}{max}[Q(4,2), Q(4,3), Q(4,4)] - Q(3,4)] \tag{4.21}$$

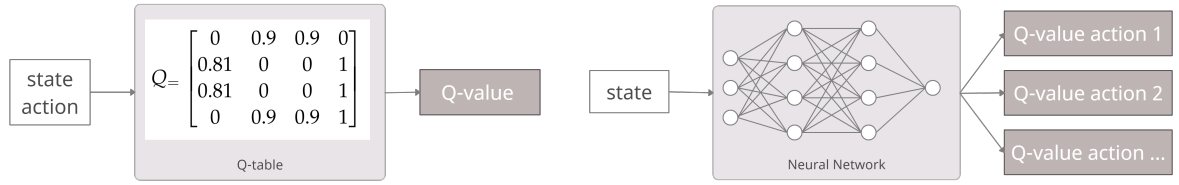
$$Q^{new}(3,4) = 0 + 0.7[1 + 0.9max[0,0,0] - 0] = 0.7 \tag{4.22}$$

The Q-matrix can be updated, the process is repeated until convergence is reached. The updated Q-matrix is shown below. Altering a program to fit the Q-function from Heinz [45] the final Q-matrix can be determined:

$$Q_{updated} = \left[\begin{array}{cccc|c} 0 & 0 & 0.63 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0.7 & 3 \\ 0 & 0 & 0 & 0 & 4 \\ \hline 1 & 2 & 3 & 4 & \\ \multicolumn{4}{c}{actions} & \end{array}\right] \quad states \qquad Q_{final} = \left[\begin{array}{cccc|c} 0 & 0.9 & 0.9 & 0 & 1 \\ 0.81 & 0 & 0 & 1 & 2 \\ 0.81 & 0 & 0 & 1 & 3 \\ 0 & 0.9 & 0.9 & 1 & 4 \\ \hline 1 & 2 & 3 & 4 & \\ \multicolumn{4}{c}{actions} & \end{array}\right] \quad states$$

In Q-learning the agent either chooses the action that gives the highest Q-value or randomly chooses an action to improve exploration. Acting upon the Q-value from any given point in the environment (in this case from any city), the agent can choose the optimum action, which will maximize the final reward.

4.2.4. Deep Q-learning



(a) Q-Learning (b) Deep Q-Learning

Figure 4.5.: (Deep) Q-Learning Structure

Q-learning uses a state and an action as the input variables. Employing the Q-matrix, the maximum Q-value is determined and set as the output (Figure 4.5(a)). [46]
Deep Q-learning uses a neural network[6] to approximate the Q-function instead of using a table or matrix to determine the Q-function. In contrast to Q-learning, only a state is used as the input value. The action, which will be taken, is determined after the network. The neural network computes all possible actions and their Q-values to the input state. The Q-values and associated actions are set as output values (Figure 4.5(b)). The action with the maximum

[6] A (artificial) neural network is a computational learning system, which is inspired by the function of neurons in the human brain. It is used for complex and high-dimensional data processing. [8]

Q-value can be chosen. Deep Q-learning is used in continuous action spaces. The downside of deep Q-learning is the non-stationary Q-matrix. Each iteration the neural network determines new actions and Q-values for the input state. [8, 40, 47]

4.2.5. Actor-Critic Algorithms

The actor-critic architecture is shown in Figure 4.6. Both policies and value-functions are learned separately. The policy structure is the actor, which selects actions. The critic estimates the value-function. It criticizes the actors' actions. [8, 40, 46]

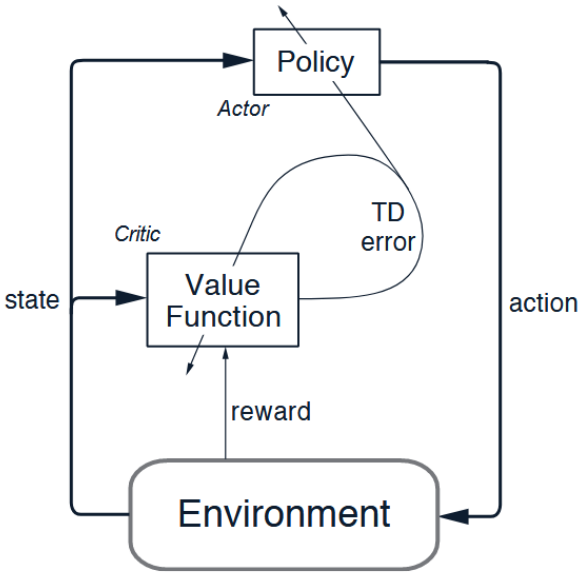


Figure 4.6.: Actor-Critic Architecture from Sutton and Barto [46]

The critic criticizes the policy and evaluates the new state after each action with the TD error δ_t. The action selected for the current state is evaluated and the value or action-value function is updated (in equation 4.23 the action-value function Q is used). The parameters of the value or action-value function are updated in equation 4.24, α_ϕ being the learning rate of the critic. Subsequently, the actor updates the policy in regard to the critics suggestions.

$$\delta_t = r_t + \gamma Q_\phi(s', a') - Q_\phi(s, a) \tag{4.23}$$

$$\phi \leftarrow \phi + \alpha_\phi \delta_t \nabla Q_\phi(s, a) \tag{4.24}$$

4.2.6. Entropy-Regularized Reinforcement Learning

In RL the term entropy is used to describe how random a random variable is. The entropy $H(x)$ can be calculated according to its probability function $P(x)$ as demonstrated in equation

4.25. Each time step the agent can get an additional reward proportional to the policy entropy at the time step. The optimal policy is therefore chanced to equation 4.26 (applying the trade-off coefficient $\alpha_{trade} > 0$).

$$H(P) = \underset{x \sim P}{\mathbb{E}}[-\log P(x)] \tag{4.25}$$

$$\pi^* = \arg\max_{a} \underset{\tau \sim \pi}{\mathbb{E}} \left[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha_{trade}\ H(\pi)) \right] \tag{4.26}$$

The value function and the Q-function change accordingly (equation 4.27 and 4.28).

$$V(s) = \underset{\tau \sim \pi}{\mathbb{E}} \left[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha_{trade} H(\pi)) | s_0 = s \right] \tag{4.27}$$

$$Q(s,a) = \underset{\tau \sim \pi}{\mathbb{E}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha_{trade} \sum_{t=0}^{\infty} \gamma^t H(\pi) | s_0 = s, a_0 = a \right] \tag{4.28}$$

The value function and the Q-function are connected by equation 4.29, while the Bellman equation for the Q-function can be seen in equation 4.30. [48]

$$V(s) = \underset{a \sim \pi}{\mathbb{E}}[Q(s,a)] + \alpha_{trade} H(\pi) \tag{4.29}$$

$$Q(s,a) = \underset{s' \sim P}{\mathbb{E}}[R(s,a,s')] + \gamma V(s') \tag{4.30}$$

4.2.7. Hyper Parameter Tuning

Hyper parameter tuning is defined as the adjustment of model design (hyper) parameters, which do not include the model parameters (which are learned during training, e.g. the loss function). Hyper parameters may e.g. include learning rate, gamma, and layers in a neural network. [49]
Every model needs a different setting of hyper parameters, thus there is no general optimum setting. Hyper parameter tuning algorithms are available, which help to automate optimizing hyper parameters. Grid search and random search can be used to determine the best hyper parameters for the model as well. [49]

4.2.8. DDPG

Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3) and Soft Actor-Critic (SAC) are hybrid algorithms, which combine the strength of Q-learning and policy gradients. The DDPG algorithm uses off-policy data and the Bellman equation to learn the Q-function and utilizes the Q-function to learn the policy. The optimal Q-function $Q^*(s,a)$ is learned by finding the optimal action (equation 4.10). The starting point to learn $Q^*(s,a)$ is the

Bellman equation (equation 4.30). From a replay buffer, a set of transitions is collected to approximate $Q_\phi(s,a)$. The Mean Squared Bellman Error (MSBE) function (equation 4.31) is set up, which indicates how close Q_ϕ satisfies the Bellman equation. d indicates whether the state is terminal and if so, no additional reward can be achieved after the current state. The goal is to minimize the MSBE loss function by using replay buffers and target networks. [50]

$$L(\phi, \mathcal{B}) = \underset{(s,a,r,s',d)\sim\mathcal{B}}{\mathbb{E}} \left[\left(Q_\phi(s,a) - (r + \gamma(1-d) \max_{a'} Q_\phi(s',a')) \right)^2 \right] \tag{4.31}$$

Experiences collected in a replay buffer are used to train a deep neural network to approximate the optimal Q-function. The replay buffer contains previous experiences. The DDPG algorithm uses two target networks to minimize the MSBE loss function. The parameters ϕ to be trained are the same as the Q-function parameters, which leads to an unstable MSBE minimization. Therefore, a set of parameters is introduced with a time delay. This new set of parameters can be considered a second network, called the target network, which uses the parameters ϕ_{targ}. The goal is to bring the Q-function as close to the target (equation 4.32) as possible.

$$y = r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \tag{4.32}$$

Once per main network update the parameter of the target network is updated by Polyak averaging (equation 4.33).

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi \tag{4.33}$$

An action, which approximately maximizes Q_{targ} is computed by the target policy network, which is established by Polyak averaging as the target Q-function. The MSBE loss is minimized by stochastic gradient descent as written in equation 4.34, with a target policy $\mu_{\theta_{targ}}$.

$$L(\phi, \mathcal{B}) = \underset{(s,a,r,s',d)\sim\mathcal{B}}{\mathbb{E}} \left[\left(Q_\phi(s,a) - (r + \gamma(1-d) Q_{\phi_{targ}}(s', \mu_{\theta_{targ}}(s'))) \right)^2 \right] \tag{4.34}$$

As the action space is continuous and the assumption that the Q-function is differentiable in regard to the action, gradient ascent can be used to learn the deterministic policy $\mu_{\theta_{\text{targ}}}$. A noise is added to the action during the training, which advocates exploration. The DDPG pseudocode in Figure 1 shows the summarized procedure of the algorithm. Two networks are randomly initialized along with the target Q-function and policy. For each state an action is selected from the policy and an exploration noise is added. The reward is received after the action is taken and a new state can be observed. The transition is stored in the replay buffer and a random batch is used to set the target. The critic is updated by minimizing the loss function and the actor policy is updated by sampling the policy gradient. As the last step, the target networks are updated. [51, 50]

Algorithm 1 Deep Deterministic Policy Gradient (DDPG) algorithm from Lillicrap, Hunt, Pritzel, et al. [50]

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathcal{B}$
for episode = 1, M **do**
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state s_1
    for t = 1, T **do**
        Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action a_t and observe reward r_t and observe new state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
        Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{B}$
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
        Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

        Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

    end for
end for

4.2.9. TD3

The Twin Delayed DDPG (TD3) algorithm is based on the Deep Deterministic Policy Gradient (DDPG) algorithm and addresses the common overestimation problem of the DDPG. As the DDPG, the TD3 algorithm is an off-policy algorithm that was developed for continuous action spaces. The main advantage of the TD3 algorithm is the double clipped Q-learning. Two Q-functions are learned at the same time by the mean-square Bellman minimization. The smaller of the two Q-functions is used to form the targets in the Bellman error loss function to avoid overestimation. The Q-function is updated more frequently than the policy and target networks and noise is added to the action target. The noise makes the exploitation of Q-function errors harder.
A clipped noise is added to each action, which are chosen from the target policy $\mu_{\theta_{\text{targ}}}$. Then the target action is clipped to be located in the defined action range. Equation 4.35 shows the target action. The Q-learning target is shaped by those actions. Target policy smoothing

prevents incorrect sharp peaks from the Q-function approximator to be exploited, which results in incorrect behavior.

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \quad \epsilon \sim \mathcal{N}(0, \sigma) \tag{4.35}$$

The two learned Q-functions use the same target, which is set by whichever Q-function provides a smaller target value (equation 4.36). Both Q-functions then regress to the same target, which helps to reduce overestimation of the Q-functions.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')), \tag{4.36}$$

$$L(\phi_i, \mathcal{B}) = \underset{(s,a,r,s',d)\sim\mathcal{B}E}{\left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2} \tag{4.37}$$

By maximizing the Q-function Q_ϕ (equation 4.38) the policy is learned.

$$\max_\theta \underset{s\sim\mathcal{B}}{\mathbb{E}}\left[Q_{\phi_1}(s, \mu_\theta(s))\right] \tag{4.38}$$

The networks are initialized with random parameters as well as the target networks. Noise is added to a selected action and the transition stored in the replay buffer. Sample transitions from the replay buffer are used to compute the target actions as well as the targets. The Q-functions are updated by gradient decent and the policy is (less frequently) updated by gradient ascent. The target networks are updated and the process repeated until a terminal state is reached. [52]

4.2.10. SAC

Model-free algorithms usually have a very high sample complexity and suffer from convergence difficulties, which results in attentive hyper parameter tuning (see section 4.2.7). The off-policy Soft Actor-Critic (SAC) algorithm maximizes entropy, while at the same time it maximizes the expected reward. The actor is acting as randomly as possible to achieve its task. SAC combines off-policy updates and an actor-critic formulation, which is stochastically stable. The maximum entropy approach improves the exploration of the environment by attaining diverse behaviors and is robust concerning model and estimation errors. There are different versions of the SAC algorithm. The current one learns two Q-functions $Q_{\phi 1}$, $Q_{\phi 2}$ and a policy π_θ. The main difference is the entropy regularization coefficient α, which is either fixed or varies over the course of the training enforcing an entropy constraint. The value function V_ψ is only learned by older versions of the SAC. [53]
The SAC algorithm is similar to the TD3, but has a few differences. SAC learns both Q-functions using the MSBE minimization, which regress to a shared target. The Polyak averaging Q-network parameters obtain the target Q-networks. The Q-network is used to

compute the shared target. It also applies the clipped double-Q trick. Unlike the TD3 the SAC algorithm uses entropy regularization. The next-state action is not applied from the target policy but instead uses the current policy. Random noise is not added, which excludes explicit target policy smoothing. Noise from stochasticity invokes the same effect as the SAC algorithm trains a stochastic policy. The entropy-regularized Q-function can be approximated by equation 4.39. The next state s' as well as the reward r are taken from the replay buffer, while the next action a' is taken from the policy.

$$Q(s,a) \approx r + \gamma(Q(s',a') - \alpha \log \pi(a'|s')) \tag{4.39}$$

Sample approximation for the target is used to set up the MSBE loss for each Q-function. The minimum Q-value is picked, using the clipped double-Q trick as in the TD3 algorithm. The loss functions are calculated by equation 4.40 using the target y in equation 4.41.

$$L(\phi_i, \mathcal{B}) = \underset{(s,a,r,s',d)\sim\mathcal{B}}{\mathbb{E}} \left[\left(Q_{\phi_i}(s,a) - y(r,s',d) \right)^2 \right] \tag{4.40}$$

$$y(r,s',d) = r + \gamma(1-d)(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s',a') - \alpha \log \pi_\theta(a'|s')) \tag{4.41}$$

The expected future return as well as the expected future entropy are to be maximized in each state by the policy. Therefore, the value function is maximized[7].

$$V(s) = Q(s,a) + \alpha H(\pi(s)) \tag{4.42}$$

$$= \underset{a\sim\pi}{\mathbb{E}} Q(s,a) - \alpha \log \pi(a|s). \tag{4.43}$$

Reparameterization is used to optimize the policy. By computing a deterministic function of independent noise, policy parameters, and the function of the state, a sample is picked from the policy. It enables to rewrite the expectation over the actions into an expectation over the noise. Therefore, the distribution is then independent of the parameters (equation 4.44).

$$\underset{a\sim\pi_\theta}{\mathbb{E}} [Q(s,a) - \alpha \log \pi_\theta] = \underset{\mathcal{N}}{\mathbb{E}} [Q(s,a_\theta) - \alpha \log \pi_\theta(a_\theta)] \tag{4.44}$$

The policy loss can be obtained by using the minimum Q-approximator. This optimizes the policy to almost the same policy optimization as the TD3 and DDPG.

$$\max_\theta \underset{a\sim\pi_\theta}{\mathbb{E}} \min_{j=1,2} Q_{\phi_j}(s,a_\theta) - \alpha \log \pi_\theta(a_\theta|s) \tag{4.45}$$

The exploration-exploitation ratio is controlled by the entropy regularization coefficient α. The higher α the more exploration will be performed by the agent. Algorithm 2 shows the Pseudocode of the SAC algorithm. The parameters, target network weights as well as the empty replay buffer are initialized. For each step in an iteration, a sample action is taken from the policy according to the current state. A sample transition is taken from the environment and stored in the replay buffer. The Q-function parameters are updated, as well as the policy weights, the entropy regularization coefficient, and the target network weights. [53]

[7]The value function uses state and action from the replay buffer.

Algorithm 2 Soft-Actor-Critic (SAC) by OpenAI [54]

Input: initialize parameters (policy parameter θ, Q-function parameter ϕ), replay buffer $\mathcal{B}$
Target parameters are set equal to main parameters $\theta_{targ,1} \leftarrow \theta_1$, $\theta_{targ,2} \leftarrow \theta_2$
repeat
    observe s, select $a \sim \pi_\theta(\cdot|s)$
    execute a
    observe s', get r and d ▷ determine if s' is terminal
    store (s, a, r, s', d) in $\mathcal{B}$
    if state s' terminal **then**
        reset environment state
    end if
    if it is time to update **then**
        for updates in range **do**
            sample $B = (s, a, r, s', d)$ from $\mathcal{B}$ ▷ random batch is sampled
            compute targets for Q-function:
            $y(r, s', d) = r + \gamma(1 - d)\left(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right)$
            update Q-functions: ▷ with gradient descent (one step)
            $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)} \left(\left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2\right) \quad i = 1, 2$
            update policy: ▷ with gradient ascent (one step)
            $\nabla_\theta \frac{1}{|B|} \sum_{(s)} \left(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s, \tilde{a}) - \alpha \log \pi_\theta(\tilde{a}|s)\right)$
            update target networks: $\theta_{targ,i} \leftarrow \rho\theta_{targ,i} + (1 - \rho)\theta_i \qquad i = 1, 2$
        end for
    end if
until convergence

4.3. Reinforcement Learning Challenges

The training of an agent in RL brings along many known challenges. This section points out some aspects to keep in mind when using RL algorithms.

Exploitation and Exploration

Exploration of the environment can be considered as discovering new information, while exploitation is defined as using already existing information to increase rewards. The exploration-exploitation dilemma is very common. The agent should exploit the currently most rewarding action, while it also should explore the environment to possibly find an even better solution. [40]

Training Stability and Reproducibility

Learning performance can be unstable in terms of reproducibility. Different runs might show different outcomes, which results in large variances difficult to compare. Thus, multiple runs for each training are necessary. The stability of the simulation model and reward rescaling improve training stability. [55]

Reward Structure

The reinforcement learning system needs manual guidance in form of reward functions, which have to be implemented by hand. Thoughts not only have to be given towards what should be rewarded but also how (in which shape). Another challenge is a reward only received at the end of an episode. The agent is taking actions at every time step, but only the final result is evaluated and rewarded. Hence, improving the policy to direct the agent in the right direction to maximize the final reward is difficult. [56]

Sample (In-)Efficiency

When a RL algorithm is data-efficient, the algorithm can use collected samples in a way to faster learn and thus, quicker improve the policy. Reinforcement learning algorithms need thousands of samples to learn a suitable policy. This problem can be tackled by parallelized learning, using multiple agents to address the same problem. [55]

Virtual Environments to Real World Model

Applying RL methods in the real world can be difficult. Tasks, which require real-world hardware, such as robotics control and autonomous vehicles, require high safety and accuracy. The exploration process of the environment is important for the agent to learn the policy. It makes a big difference if this exploration process happens in a simulated environment or high complexity of the real world. The simulation environment might not replicate all

the physical constraints of the real world. A sim-to-real transfer[8] might cause difficulties when the simulated environment does not correspond to the real-world environment. This problem is called the reality gap. An approach to avoid the simulation to reality gap is, to train the agent in reality instead of a simulation environment. In complex systems, such as rocket engines, this approach is not applicable as too many failed attempts and abrupt engine shutdowns lead to component and system damage. [55]

4.4. Summary

Reinforcement Learning is a form of machine learning, in which an agent takes actions in an environment, according to a policy, to maximize a reward. Reinforcement Learning (RL) algorithms can be categorized into model-free and model-based algorithms. The agent learns the model by exploring the environment in model-free RL. During the training, the policy can be updated for policy optimization. The Q-matrix determines the expected reward for the next state at a given state-action pair. Introducing a neural network for deep Q-learning enables Q-learning in continuous action spaces. [40, 8]
The Deep Deterministic Policy Gradient (DDPG) algorithm is an off-policy algorithm, which is used for environments with continuous action spaces. It used deep Q-learning and initializes two networks to optimize Q-function and policy. [51, 50]
The overestimation problem of DDPG is addressed in the Twin Delayed DDPG (TD3), which is based on DDPG. It uses clipped double Q-learning, in which the smaller of the two Q-values is used. The policy is updated less frequently than the Q-function. To avoid the exploitation of Q-function errors, target policy smoothing is applied. [52]
The Soft Actor-Critic (SAC) algorithm is similar to TD3 but maximizes entropy regularization to improve exploration of the environment, which improves robustness regarding model estimation errors. [53]
The optimum algorithm parameter settings vary for each problem and have to be customized by hyper parameter tuning. [49]
The training of a RL agent brings along many challenges, such as the exploitation-exploration dilemma and training stability, and the ability to reproduce training results. The reward structure needs to be well thought through and sample (in-)efficiency can be solved by using multiple agents to address the same problem. The application of an agent, trained in a simulated environment, in the real world might cause problems, due to the reality gap. To avoid the reality gap the simulation environment must represent the real-world environment as close as possible, or the agent could be trained in the real world instead of using a simulation environment. [40, 55, 56]

[8] simulation model to real-world transfer of the training

5. LUMEN Implementation in Reinforcement Learning

LUMEN is a bread-board engine, with six controllable valves to reach different operation points. Combustion chamber pressure and Mixture Ratio (MR) targets are to be met while staying within the engine's boundary conditions. The goal is to reach the operation points without unnecessary peaks, which could damage engine parts. Besides, propellant usage should be minimized to optimize the performance of the engine. Six different controllable valves can be adjusted to reach the operation points.

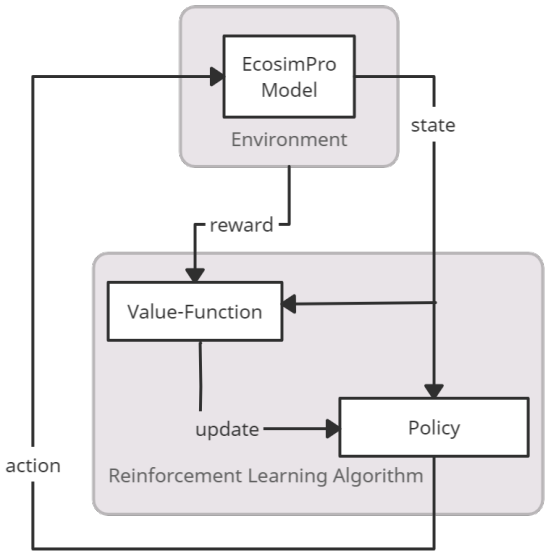


Figure 5.1.: Implementation of LUMEN in RL

The EcosimPro LUMEN model from section 3.3 is used to train an agent in RL.
DDPG, TD3 and SAC algorithm frameworks already exist in several open-source libraries. The open source framework ray enables parallel calculations, which shortens computational time. RLlib is an open-source library for reinforcement learning, supporting Pytorch, which is used to implement the algorithm. RLlib forms all data interchange into sample batches, which form a trajectory. Batches are collected by RLlib from rollout workers, which collects training data from the simulation. Each worker calls an EcosimPro model (the deck) and uses the simulation to retrieve data. The Python agent calculates the reward regarding the

observations, coming from each rollout worker. It subsequently chooses an action, which is then implemented by the rollout worker. When a done signal is received (e.g. a time limit of a 20 s run), the episode is terminated. [57]
The EcosimPro engine model is exported into a deck, a standalone application, in which pre-defined input and output variables are accessible [35].
The deck receives predefined variable values, input values, performs one timestep, and returns the output variables, which were calculated on the base of the EcosimPro equations (Figure 5.1). Output variables are predefined, such as combustion chamber pressure and MR. The reinforcement learning agent uses the output variables as observation space variables, which are only observed to determine the state. The reward for the last action taken regarding the current state is determined. The input variables are set as the action space, which the agent is supposed to adjust to reach an optimum state. The agent modifies the action space variables according to its policy and transmits them as an input value to the EcosimPro model. This process is repeated until a terminal state (either the optimum state or end of an episode) is reached.
During the training the SAC algorithm proved to be the most stable out of the three choices. Hence, the following displayed results are computed using SAC.
The original SAC configuration is customized:

```
buffer_size = 100000
learning_starts = 256
timesteps_per_iteration = 5000
target_entropy = auto
gamma = 0.9
```

The detailed SAC algorithms settings are displayed in Table A.1.

5.1. Reinforcement Learning Set-Up

The agents objective is to maximize the total reward. The total reward (equation 5.1) is set up by multiple rewards rew_i and a penalty. The penalty comes into action if a constraint is violated, e.g. minimum LNG injection temperture or minimum cooling channel pressure (see Table 3.2).

$$rew = \sum rew_i + Penalty \tag{5.1}$$

Constraint violations are punished by a penalty. Penalties implemented as the following code:

```
if variable < constraint-value-minimum:
    Penalty = -2
else:
    Penalty = 0
```

The rewards are implemented to lead the agent in the right direction of desired values (such as combustion chamber pressure and MR). Reward function shaping is important to manipulate

the given reward. Using an exponential or root function may lead the agent towards the desired value. Only giving out a reward when the exact value is reached (plateau) may be counterproductive, as the agent is only able to find this specific value by chance [58]. In this case a negative root function is selected (Figure 5.2). The scaled reward function for combustion chamber pressure, mixture ratio, cooling channel mass flow rate, and cooling channel pressure can be seen in equation 5.2. The reward function consists of the derivation from the setpoint divided by a scale factor ϵ, to the power of the exponent.

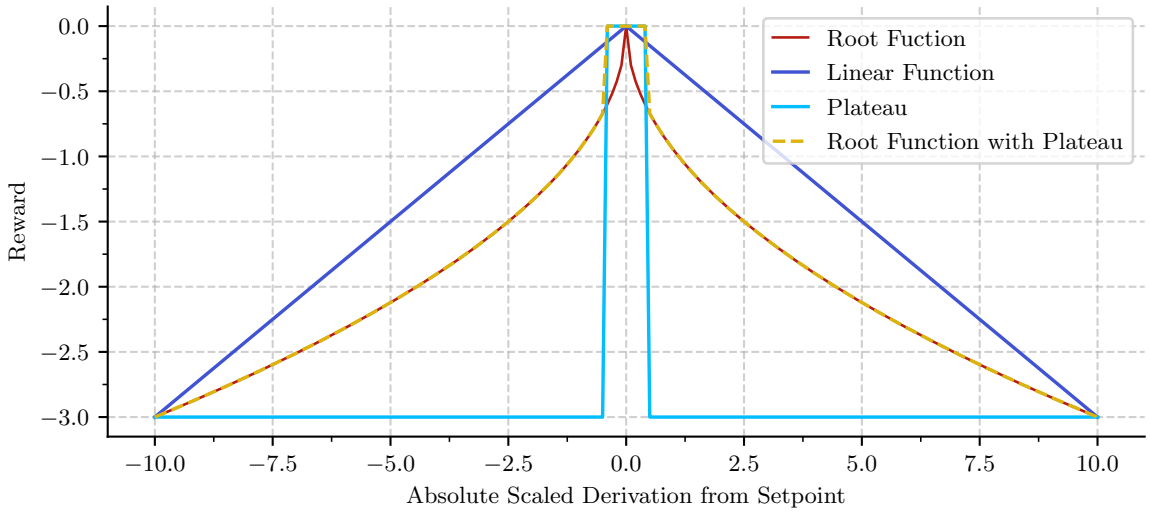


Figure 5.2.: Reward Function Shapes

$$rew_X = -\left[\frac{|X - X_{target}|}{\epsilon_X}\right]^{exponent} \tag{5.2}$$

$$rew_X = -\frac{X}{\epsilon_X} \tag{5.3}$$

A linear reward function as seen in equation 5.3 is implemented when no specific target value is defined. This equation can be used if a variable is to be minimized, e.g. propellant usage. Table 5.1 shows the applied parameters for the different reward functions. The exponent is

ϵ Combustion Chamber Pressure	5
ϵ Mixture Ratio	0.5
ϵ Cooling Channel Mass Flow Rate	0.5
ϵ Cooling Channel Pressure	10
ϵ Bleed Mass Flow Rate	1
Exponent	0.5

Table 5.1.: Reward Function Parameters

set to 0.5 for all reward functions.
The control valves are set as the action space variables, while combustion chamber pressure, LNG injection temperature, mixture ratio, cooling channel pressure, cooling channel mass flow rate and LNG and LOX turbine pressure and temperature are defined as observation space variables. The combustion chamber wall temperature should be limited to 900 K. Since the simulation of the chemical reaction in the combustion chamber is not modeled accurately, the combustion chamber wall temperature constraint is not implemented as a RL constraint, as the empirical correlation for heat transfer in the cooling channel for LNG is not accurate enough. The applied cooling channel mass flow rates are from the preliminary design and resulted in acceptable combustion chamber wall temperatures and thus, cooling channel wall temperatures.

5.2. Combustion Chamber and Mixture Ratio Control

After each episode during the training a checkpoint is generated, which can be evaluated. During the course of the training, the policy is improved, which subsequently improves the results. A comparison of early checkpoint results and a higher checkpoint can be seen in Figures 5.3 to 5.4.
The valve positions and targets to be met can be seen in Figure 5.3. The combustion chamber pressure and mixture ratio from the operation points are set as target values. At the beginning of the training (checkpoint 2), the agent was not able to meet combustion chamber pressure and MR targets. After the training progresses (checkpoint 44), it is now able to meet both setpoints.
The LNG injection temperature constraint can be seen in A.1. Pump outlet pressure and turbine inlet pressure have boundary conditions according to table 3.2. If the agent crosses the minimum or maximum value a penalty will be received.
Figure 5.4 shows the agent receiving a penalty, after the minimum LNG injection temperature constraint is violated. When the penalty is received, all other rewards are automatically set to 0. Later in the training, after 44 checkpoints, the agent can meet all targets and maximizes its reward. The total weighted cumulative reward indicates, how fast the agent is able to meet the target values. The lower the cumulative reward, the better. Equation 5.1 can be seen as the total cumulative reward function. All scaled rewards, as well as penalties, are summed up. The different elements of the reward function and its total sum are displayed in Figure 5.4.

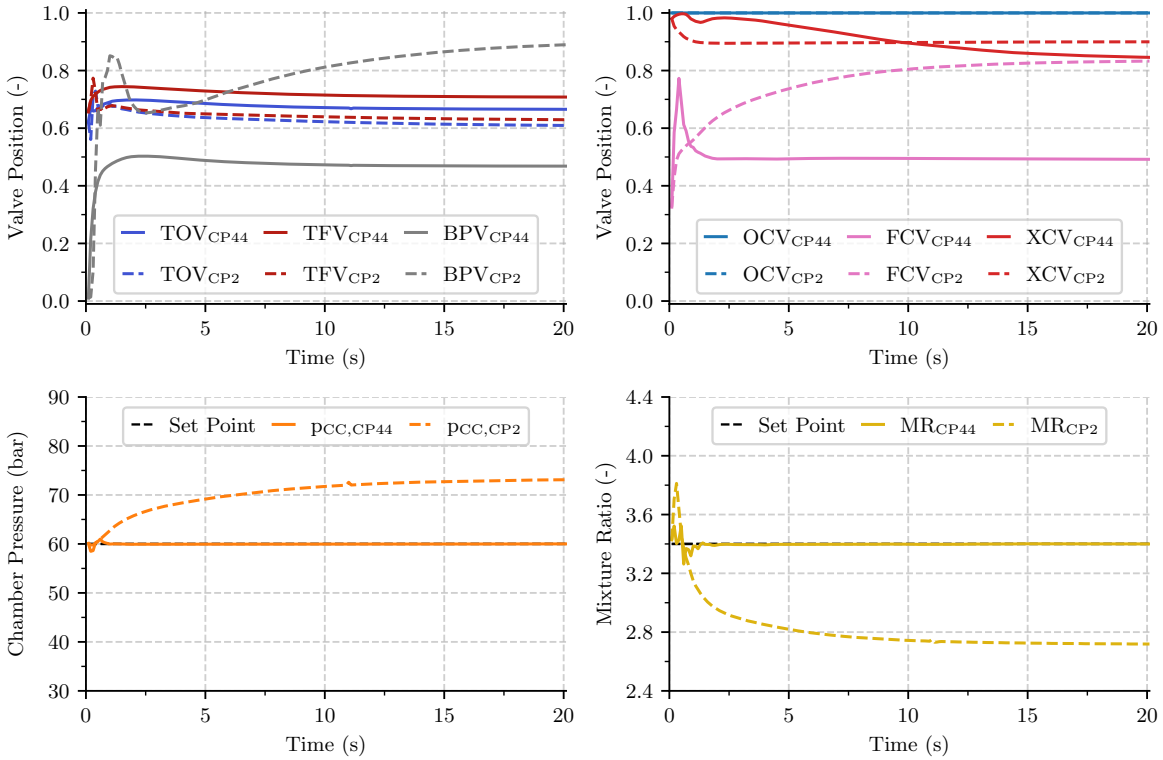


Figure 5.3.: Checkpoint Comparison Target Values: (–) CP 2, (- -) CP 44

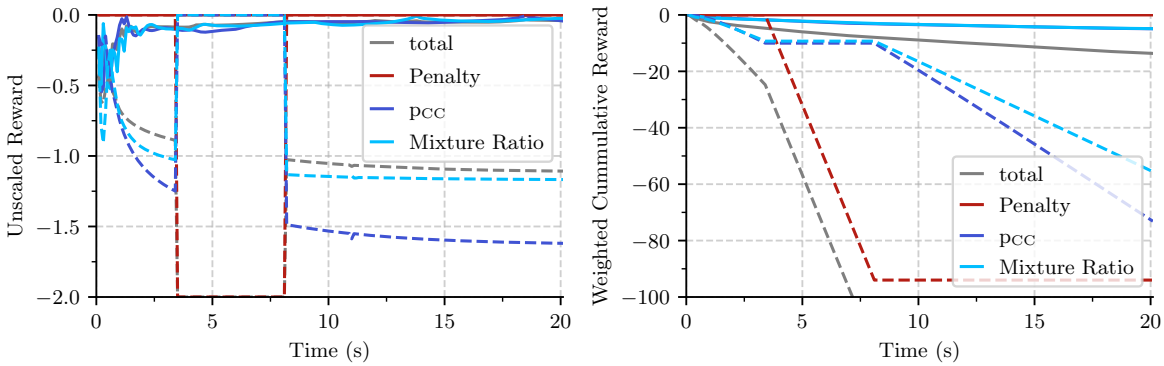


Figure 5.4.: Checkpoint Comparison Rewards: (–) CP 2, (- -) CP 4

5.3. Cooling Channel Mass Flow Rate Control

The cooling channel mass flow rate influences the LNG injection temperature as well as the inlet temperatures of LNG and LOX turbine, which determines their performance. Therefore, cooling channel mass flow rate is an important variable and one goal of optimal engine

control is, to control the cooling channel mass flow rate precisely for optimal performance and durability.
To reduce fuel wastage BPV is fixed at a fully closed position. To limit valve actions OCV is fixed at a fully opened position. At the same time, performance is maximized, as no additional pressure loss occurs. The cooling channel mass flow rate from $\dot{m} = 1.4\,\mathrm{kg\,s^{-1}}$ to $2.7\,\mathrm{kg\,s^{-1}}$ is regulated.

5.3.1. Fixed BPV, Fixed OCV

Setting BPV and OCV as fixed valves the cooling channel mass flow rate is regulated from $\dot{m} = 1.4\,\mathrm{kg\,s^{-1}}$ to $2.7\,\mathrm{kg\,s^{-1}}$ at $p_{CC} = 60\,\mathrm{bar}$ and $MR = 3.4$. Figure 5.5 displays the derivation from the target values of combustion chamber pressure, MR and cooling channel mass flow rates at different cooling channel mass flow rates. In addition, the cumulative total reward and the cumulative reward for the different setpoint variables are shown. In general, reaching the target values works well. However, high cooling channel mass flow rates show larger derivations (up to 14 % combustion chamber pressure derivation).
High mass flow rates are difficult to reach without using BPV, as the LNG injection temperature constraint is violated when the cooling channel temperature drops. The turbine inlet temperate drops subsequently, which results in decreasing pump power and thus decreasing combustion chamber pressure. The agent meets the cooling channel mass flow rate setpoint before meeting the set combustion chamber pressure. Implementing a condition, which primarily regulates combustion chamber pressure and MR before concerning other set values causes the agent to meet fewer set points. Figure 5.5 shows the decreasing total reward, which is received at higher cooling channel mass flow rates. For a cooling channel mass flow rate of $2.6\,\mathrm{kg\,s^{-1}}$ and $2.7\,\mathrm{kg\,s^{-1}}$, mixture ratio and cooling channel mass flow rate targets are met with less than 6 % derivation, resulting in decreasing combustion chamber pressure and worse reward.
Low cooling channel mass flow rates ($\dot{m} = 1.4\,\mathrm{kg\,s^{-1}}$ and $1.5\,\mathrm{kg\,s^{-1}}$) are achievable without using BPV, but result in a cooling channel wall temperature above 900 K. Hence, this operation point would not be applied to the real-world model. When the combustion chamber wall temperature rises, the cooling channel fluid heats up more, and thus the turbine inlet temperature increases, which leads to higher pump outlet pressure. The cooling channel pressure rises.
The total cumulative reward leaving BPV closed and OCV fully open is in the range from -112 to -30.
Lowering the cooling channel pressure leads to a lower turbine inlet pressure and thus lower pump outlet pressure. The combustion chamber pressure decreases. Thus, decreasing the cooling channel pressure will help to regulate combustion chamber pressure, while keeping up with other targets, hence, in the next step BPV is implemented into the action space and set as an adjustable valve. Increasing the number of adjustable valves may lead to operation point expansion.

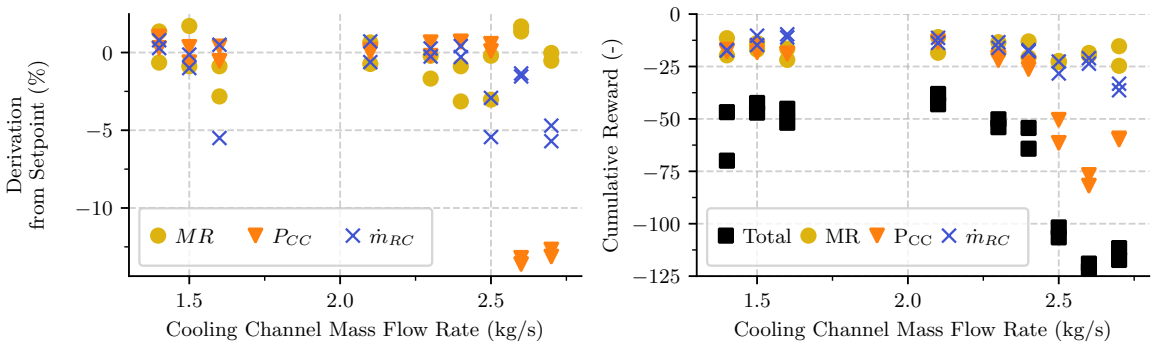


Figure 5.5.: Setpoint Derivation and Cumulative Reward (fixed BPV, fixed OCV)

5.3.2. Adjustable BPV, Fixed OCV

Higher cooling channel mass flow rates are achievable when BPV is adjustable. Figure 5.6 shows setpoint derivation of less than 3 % for all cooling channel mass flow rates from 1.4 kg s^{-1} to 2.7 kg s^{-1}.
As BPV is adjustable, the cooling channel pressure can be further regulated. Using BPV enables further decoupling of cooling channel mass flow and turbine mass flow. Thus, lower combustion chamber pressure is achievable.
Mass flow rates 2.5 kg s^{-1}, 2.6 kg s^{-1} and 2.7 kg s^{-1} are reached with less than 1 % derivation from setpoints. The bleed mass flow rate downstream BPV reaches approximately 0.5 kg s^{-1}, resulting in the highest fuel wastage of the displayed calculations.
The total cumulative reward adjusting BPV, leaving OCV fully open is in the range from −41 to −11.

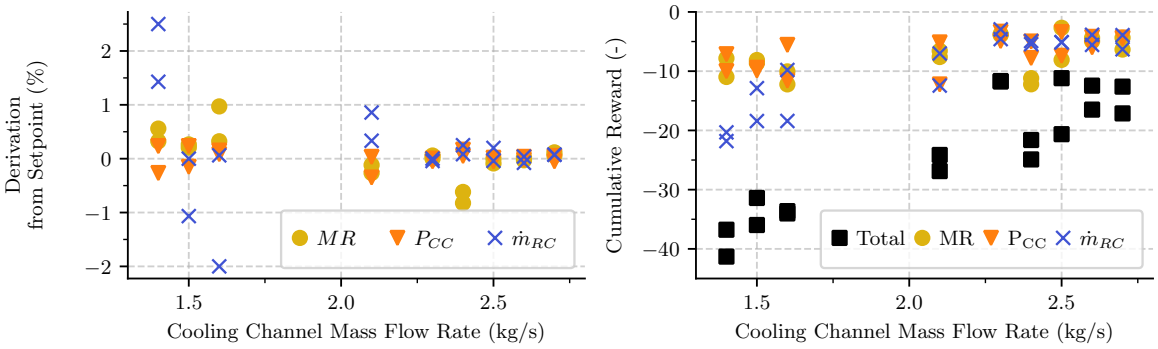


Figure 5.6.: Setpoint Derivation and Cumulative Reward (adjustable BPV, fixed OCV)

5.3.3. Adjustable BPV, Adjustable OCV

Implementing BPV and OCV as adjustable valves, results in high variation of solutions. As the agent is able to adjust more valves, the training gets more difficult. OCV is closed to less

than 0.8 % of the full opening position to decrease the combustion chamber pressure for high cooling channel mass flow rates, resulting in higher total cumulative rewards.
Cooling channel mass flow rates $1.4\,\mathrm{kg\,s^{-1}}$ and $1.5\,\mathrm{kg\,s^{-1}}$ receive lower total cumulative rewards, as the agent uses its new degree of freedom to close OCV to approximately 0.9 %. The agent takes longer to reach the cooling channel mass flow rate target, which leads to lower cumulative rewards.
The total cumulative reward adjusting BPV and OCV is in the range from -59 to -13. Even though for this operation point including OCV seems to be not necessary, OCV needs to be adjusted to reach lower combustion chamber pressures (e.g. 35 bar).

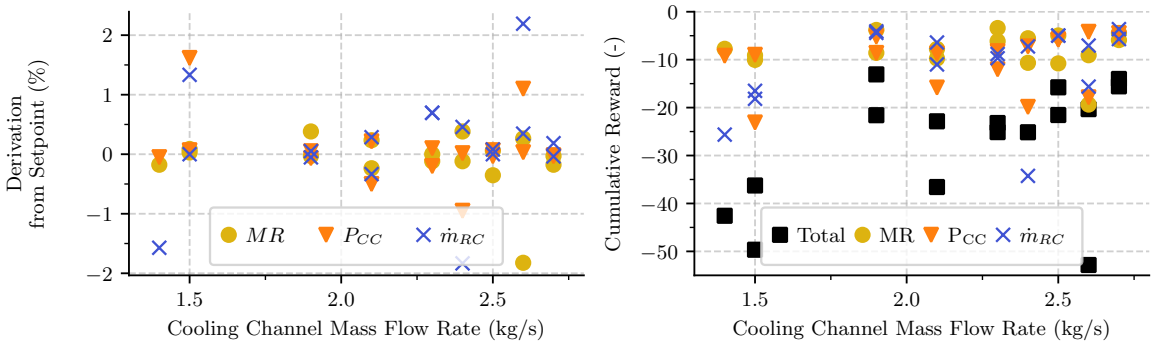


Figure 5.7.: Setpoint Derivation and Cumulative Reward (adjustable BPV, adjustable OCV)

5.4. Cooling Channel Pressure Control

As the system is controlled by six adjustable valves, it is under-determined. Thus multiple valve position combinations are possible for each setpoint. To avoid multiple solutions an additional target value (the cooling channel pressure) can be implemented. The cooling pressure channel reward is added as a mixture of a plateau and root function (Figure 5.2). The plateau is called the corridor. When the cooling channel pressure value lies within the corridor, the maximum reward of 0 is received. The root function reward shape leads the agent towards the maximum reward. The implementation into python can be seen below. The cooling channel pressure reward equation has the shape of the other reward functions seen in equation 5.2.

```
if abs(P_RC - P_RC_setpoint) < Corridor:
    rew = 0
else:
    rew = Reward_Equation_P_RC
```

The corridor was set to 10 bar, which gives a window of 20 bar, in which the maximum reward of 0 is received. The cooling channel pressure target is used to control the pressure during operation point transition in section 5.5 (Figure 5.9).

5.5. Operation Point Transition (Throttling)

In this section, different operation points are to be reached to analyze the dynamic system behavior during operation point transition. Including an adjustable BPV various solutions for the same operation point are possible. To avoid an under-determined system BPV is fully closed and OCV is fully open, leaving four adjustable valves and four setpoint variables (combustion chamber pressure, MR, cooling channel pressure, and cooling channel mass flow rate). Transition from OP1 (60 bar, 3.4) to OP2 (80 bar, 3.4) to OP8 (80 bar, 3.8) can be seen in Figure 5.9 and is presented in Dresia, Waxenegger-Wilfing, Santos Hahn, et al. [31]. Combustion chamber pressure, MR and cooling channel mass flow rate targets are met and the cooling channel pressure remains within its boundaries. Peaks are determined at transition points. The agent's task is to reach the new target as quickly as possible to maximize the reward. In reality, those peak values are not desirable, as they might cause damage. The agent sees it more fit, to quickly adjust all values to collect the maximum reward. A reward function could be implemented to prevent peaks for future computations.
In addition, quick MR change is not possible in real life models, as the combustion process cannot adapt as quickly as displayed to MR changes.
Starting from the sequence of Figure 5.9 at 50 s, the engine shall be throttled down to a lower combustion chamber pressure, between 40 bar to 70 bar, while decreasing the MR to 3.0 at the same time. Cooling channel pressure is targeted at 100 bar with a corridor of 10 bar, which sets the acceptable cooling channel pressure range between 90 bar to 110 bar. Combustion chamber pressure, MR, cooling channel pressure and cooling mass flow rate targets are met in all calculations.
Operation points at p_{CC} =35 bar cannot be reached with fixed BPV and OCV positions. The cooling channel pressure cannot be lowered below 46 bar, the pressure cannot fall below the critical point of methane. BPV cannot be used to adjust pressure behind the cooling channel, which leads to a high fuel injection pressure. To reach operation points at p_{CC} =35 bar, BPV and OCV have to be set as adjustable valves.

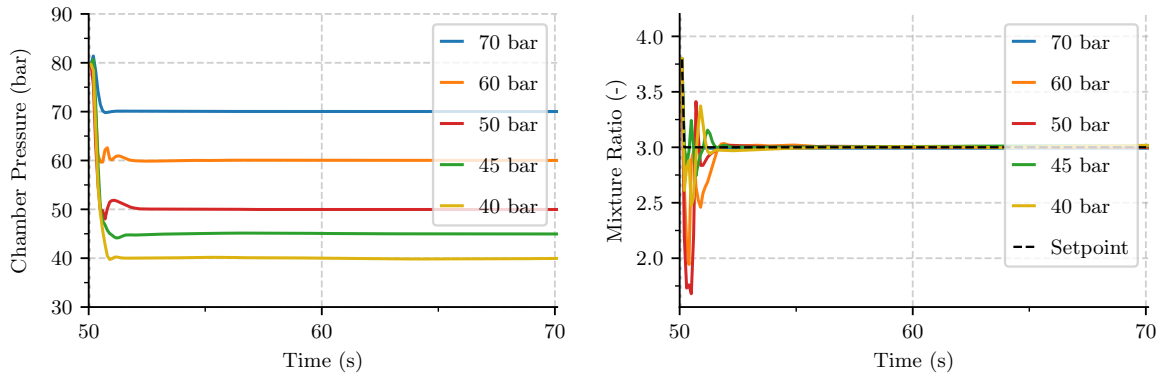


Figure 5.8.: Operation Point Transition

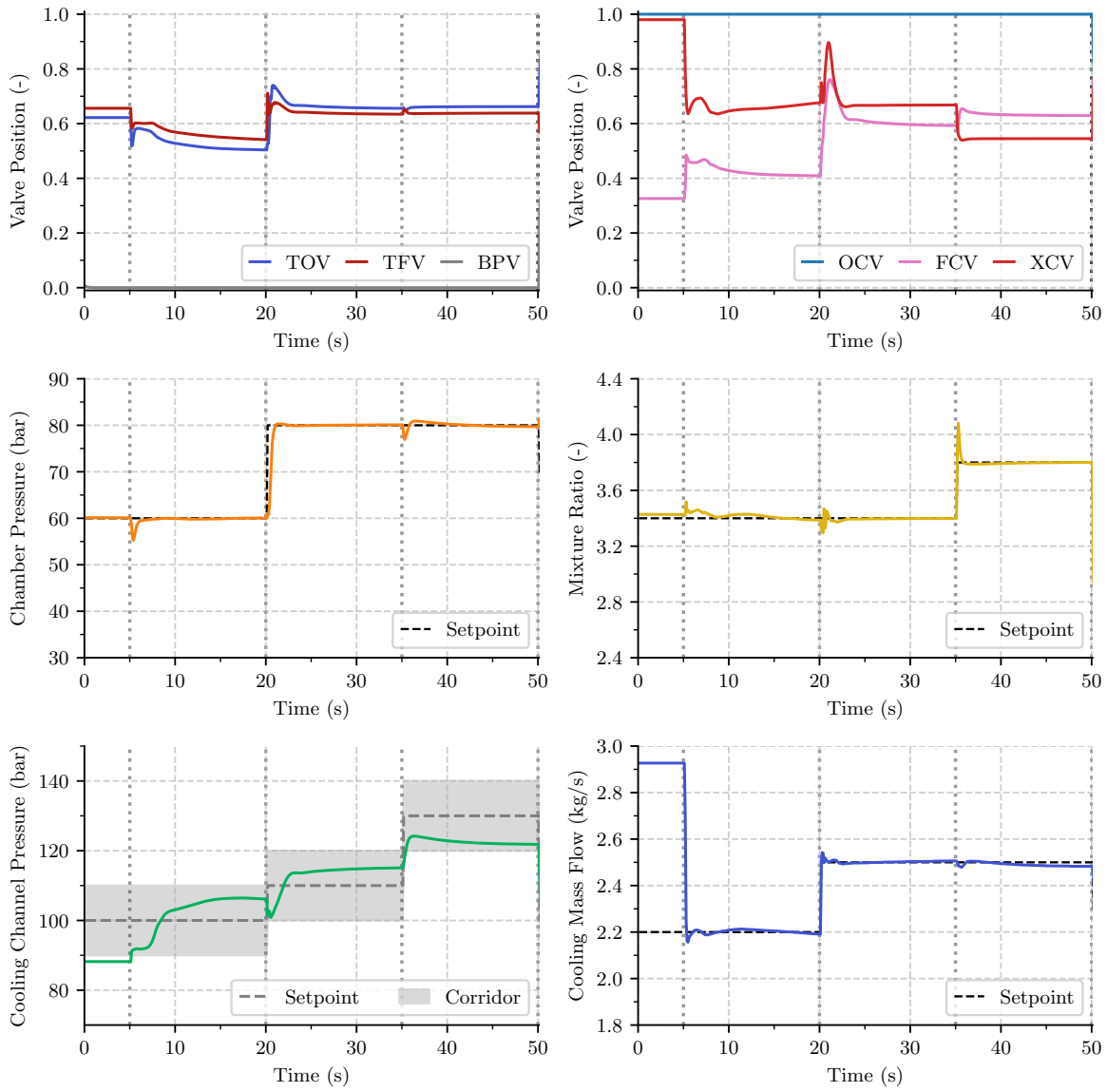


Figure 5.9.: Operation Point Transition (OP1 → OP2 → OP8)

5.6. Optimization (Minimizing Bleed Mass Flow Rate)

Table 5.2.: Bleed Mass Flow Rate with (2) and without (1) Minimizing Bleed Mass Flow Rate Reward Function

		Cooling Channel Mass Flow Rate [kg/s]					
	Variable	1.4	1.5	1.6	1.9	2.0	2.1
Total Bleed Flow	(1) $\dot{m}_{Bleed}$ [kg/s]	1.04	1.04	1.14	1.37	1.42	1.51
	(2) $\dot{m}_{Bleed}$ [kg/s]	0.94	0.90	0.98	1.02	1.03	1.07
	Derivation [%]	9.88	13.03	13.90	25.37	28.02	29.31
Total LNG Flow	(1) $\dot{m}_{\text{total LNG}}$ [kg/s]	2.76	2.79	2.85	3.09	3.15	3.23
	(2) $\dot{m}_{\text{total LNG}}$ [kg/s]	2.66	2.62	2.70	2.75	2.75	2.79
	Derivation [%]	3.63	4.50	5.29	10.90	12.77	13.94

	Variable	2.2	2.3	2.4	2.5	2.6	2.7
Total Bleed Flow	(1) $\dot{m}_{Bleed}$ [kg/s]	1.56	1.61	1.42	1.66	1.69	1.75
	(2) $\dot{m}_{Bleed}$ [kg/s]	1.09	1.09	1.21	1.16	1.19	1.24
	Derivation [%]	30.45	32.21	14.86	30.08	29.71	29.31
Total LNG Flow	(1) $\dot{m}_{\text{total LNG}}$ [kg/s]	3.28	3.33	3.15	3.38	3.41	3.46
	(2) $\dot{m}_{\text{total LNG}}$ [kg/s]	2.85	2.81	2.84	2.88	2.92	2.96
	Derivation [%]	12.99	15.57	9.98	14.87	14.55	14.66

The engine's performance shall be optimized, to make LUMEN as efficient as possible. To reach optimal engine control, I_{sp} could be maximized or rather propellant utilization minimized. Minimizing propellant usage can be achieved by reducing fuel dumping. To reduce fuel wastage, not only by opening BPV but also through the turbines, an additional reward function to minimize the total bleed flow is introduced. The Bleed Reward Function (BRF) adds an additional, linear reward component as in equation 5.3, to minimize the total bleed flow. As BPV does not have to be fully closed the agent now can reduce $\dot{m}_{BPV}$ by opening XCV. BPV is adjustable, while OCV stays fully open.
Table 5.2 shows the reduced $\dot{m}_{Bleed}$ and total $\dot{m}_{\text{total LNG}}$, when the new reward function is implemented, in comparison to the reward function that only uses the derivations from the target setpoint.
The total LNG flow can be reduced by 4 % to 16 %. This new reward function successfully minimizes the total bleed mass flow rate (turbine and BPV dump) and thus reduces fuel wastage by 9 % to 32 %, which improves the engines I_{sp}, and thus the necessary propellant mass. In real-life applications reducing fuel dumping can result in longer missions life spans, as the fuel availability is prolonged. As an alternative, the same mission needs less fuel, which reduces propellant weight and thus leaves capacity for more heavy payloads.
Figure 5.10 visualizes the comparison of a cooling channel mass flow rate of 2.1 $\text{kg}\,\text{s}^{-1}$ achieved with and without the minimizing bleed mass flow rate reward function. Combustion chamber pressure, MR and cooling channel mass flow rate targets are met for both calculations. Using

the BRF, the agent closed BPV to less than 0.2 opening position. The BPV bleed mass flow rate is reduced to almost 0.0 kg s^{-1}. TFV and TOV are slightly closed to maintain the turbine mass flow rate. As the cooling channel pressure rises when BPV is closed, the turbine inlet temperature rises. FCV is closed and XCV opened to maintain the combustion chamber pressure. When the BPV-bleed is reduced by the reward function, the cooling channel pressure rises, which causes the LNG injection temperature to increase as well. The total bleed mass flow rate is reduced by 29 %, while the total LNG mass flow rate is reduced by 14 %.

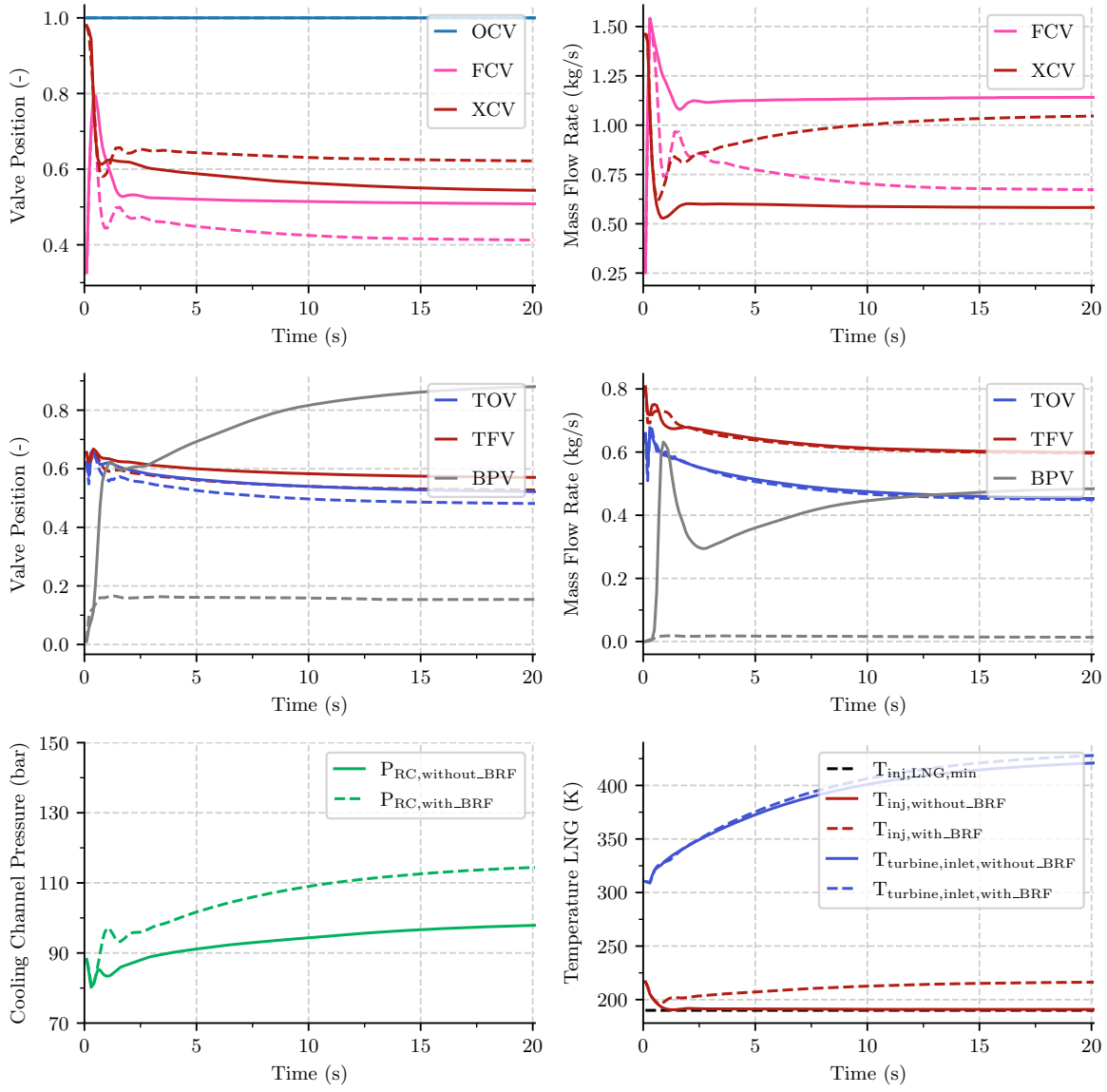


Figure 5.10.: OP 1 with (–) and without (-) Minimizing Bleed Reward Function at $\dot{m}_{RC} = 2.1 kg/s$

5.7. Robustness of Reinforcement Learning Control

RL works well for complex models in an isolated simulated environment. However, it is difficult to transfer the trained agent onto a real-world model. The trained model is vulnerable to external disturbances, such as ambient pressure and temperature changes, valve malfunctioning, or mechanical failures and needs to be able to handle delayed valve actions and sensor data. [59, 9]
The engine heats up during the first few seconds of performance, which causes the physical properties of mechanical elements to vary. Besides, the trained model is specialized in performing a predefined task, and not trained for all tasks the agent might be asked to perform.
Another aspect, which has to be taken into account is that the simulated engine can recover from setpoints, which would be impossible for a real-life engine to recover from. The EcosimPro model has modeling limits and thus only represents the LUMEN engine. The simulation model might differ compared to the real-world model.
To increase robustness domain randomization can be applied during training. A parameter is randomly modified during the simulation, which helps the agent to become resistant to the trained parameter variation. This method supports reducing system inaccuracy and thus helps to transfer the trained model onto a real-life engine. [60]

5.7.1. Impact of Different Initial States

To determine whether the training is stable towards changing initial conditions, the starting point is changed to different combustion chamber pressures and MRs than the ones trained with. The initial valve settings are changed to fit different initial combustion chamber pressure or valve settings. The initial position is run for 15 s before the agent applies the trained policy to achieve steady-state conditions. An initial state, which does not violate any constraints is essential to meet all approximate targets within a few seconds.
Starting from different combustion chamber pressures and MRs are not a problem for the agent. The agent can recover quickly from the unexpected starting points, meeting combustion chamber pressure and MR, as well as cooling channel mass flow rate and pressure targets. Setting the initial valve positions to 1.0, the agent needs slightly less than 5 s to adjust the cooling channel mass flow rate, while all other targets are already achieved. This is mainly caused by the time it takes to close BPV from its fully open position. For all initial settings, the agent can find the approximate target value within 20 s.
When all valve positions are initially set to 0.1 the agent has difficulties reaching the target values, especially MR. The low valve settings cause constraint violations, e.g. LNG injection temperature and turbine inlet pressure. As the valves are almost fully closed, the turbine inlet pressure cannot be maintained, which causes a penalty. As the agent receives penalties within the first seconds, it needs time to recover and find the target values. When all valves positions are fully open at the initial state, the agent can meet all targets earlier and without highly fluctuating MR.
The valve position settings of 0.1 for valve positions besides BPV are not realistic, as the mass

flow rate is limited and would cause engine shutdown. This example shows the sim-to-real gap, as the agent can meet all targets when in reality this initial setting would cause engine failure.

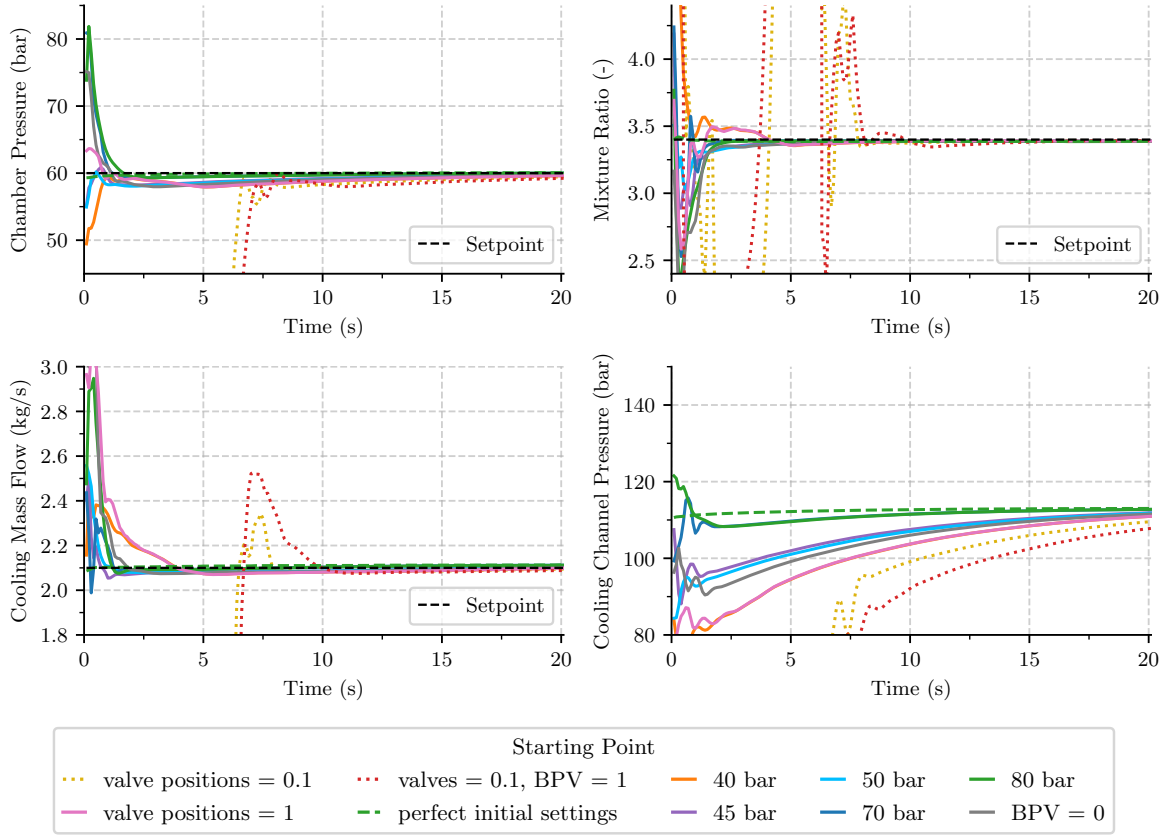


Figure 5.11.: Initial Starting Point Comparison

5.7.2. Impact of Sensor Noise

In RL there are two types of adding noise to the training. When noise is directly implemented into the parameters (in this case: combustion chamber pressure, MR, ...), the noise is defined as parameter space noise. Whereas action space noise is injected into the action before it is taken. This noise is implemented in this section. When noise is implemented into the training, it helps the agent's exploration of the environment. [61]
Here, it is added to the evaluation to examine how stable the agent's behavior is after the training.
Random average noise is set (0.5 %, 1 % and 2 %) and added to the sensor, which detects the valve position. Thus the valve position picked by the agent is not constant but manipulated by an added noise. This simulates derivation from the actual valve position the agent selects, as real life sensors are not 100 % accurate and show noise.

Displayed in Figure 5.12, fluctuations around the target values occur and grow as expected as the added average noise increases. As the sensors of the valve positions are randomly manipulated the targeted values vary. The agent manages to counteract the valve positions according to the derivation of the target values and follow the paths of 0 % noise.
As the noise grows, the fluctuations expectantly increase. For a noise 0 % to 1 % the agent gets acceptable target values. The target values of 2 % noise fluctuate significantly. The agent is robust against evaluation noise up to 2 %.

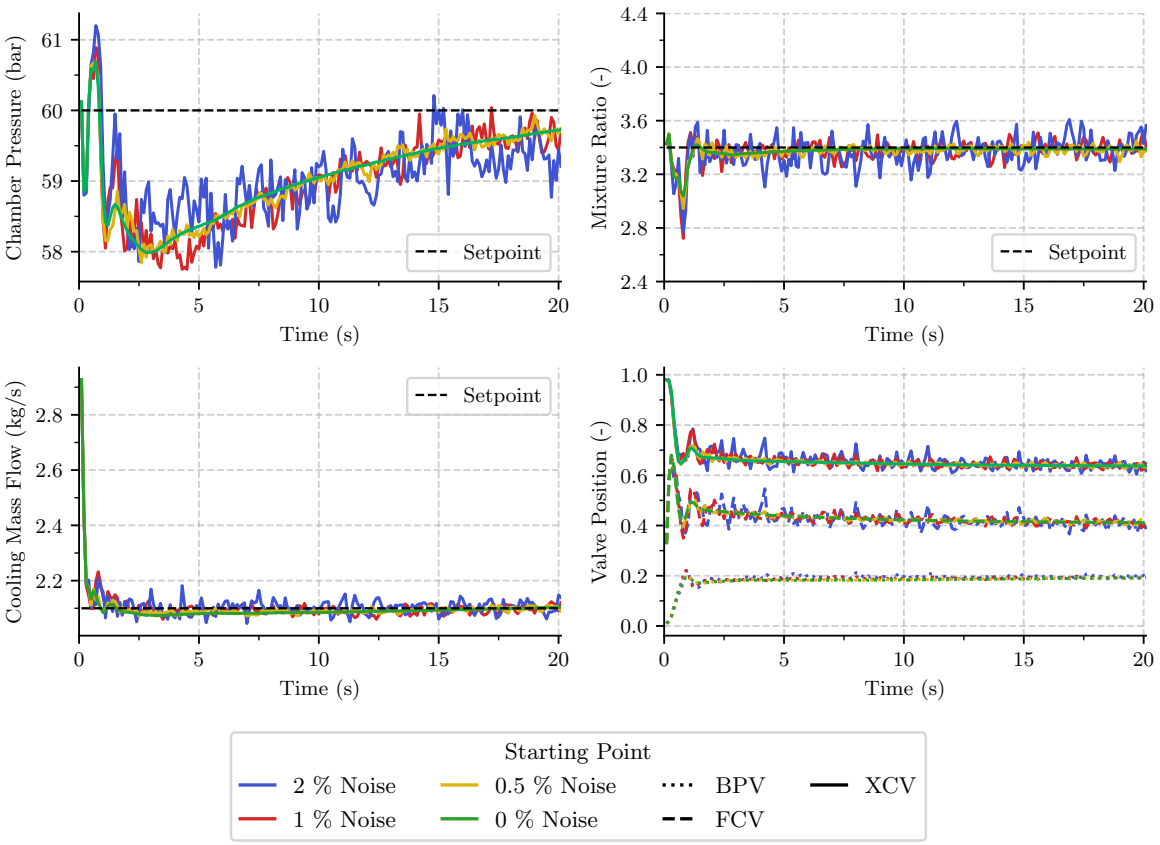


Figure 5.12.: Valve Noise Comparison at $\dot{m}_{RC} = 2.1 kg/s$

5.7.3. Impact of Parameter Change after Training

The operating stability of the engine depends on the stability of the valve positions. If the valve position fluctuates, the target values fluctuate as well. The agent learns, how fast system variables change when a valve position is adjusted.
τ, the valve opening time constant, is set in the EcosimPro model.
Figure 5.13 shows a comparison between $\tau = 0.2$, the value set during training; and $\tau = 0.1$, a value adjusted after training. Lowering the time constant τ results in fluctuation of valve position and thus system variables. The trained agent is not able to adapt to quickly changing

system variables, such as combustion chamber pressure and MR. As MR overshoots its target value, due to the quicker valve adjustment, the agent counteracts by changing the valve position again in the next step. The system is given no time to even out and valve and system variable oscillation can be observed.
Increasing τ causes the valves to react delayed in comparison to what the agent has learned. The system variables values even out over time. The valves are adjusted to meet the target variables, but the system takes longer to reach them. Visualization can be seen in A.2.
Change of system parameters after the training is difficult. The agent is not able to adapt to new system parameters as it only acts upon the trained policy. Hörger [60] uses domain randomization to train the agent for system parameter change, which could prevent valve fluctuation due to parameter changes after the training.

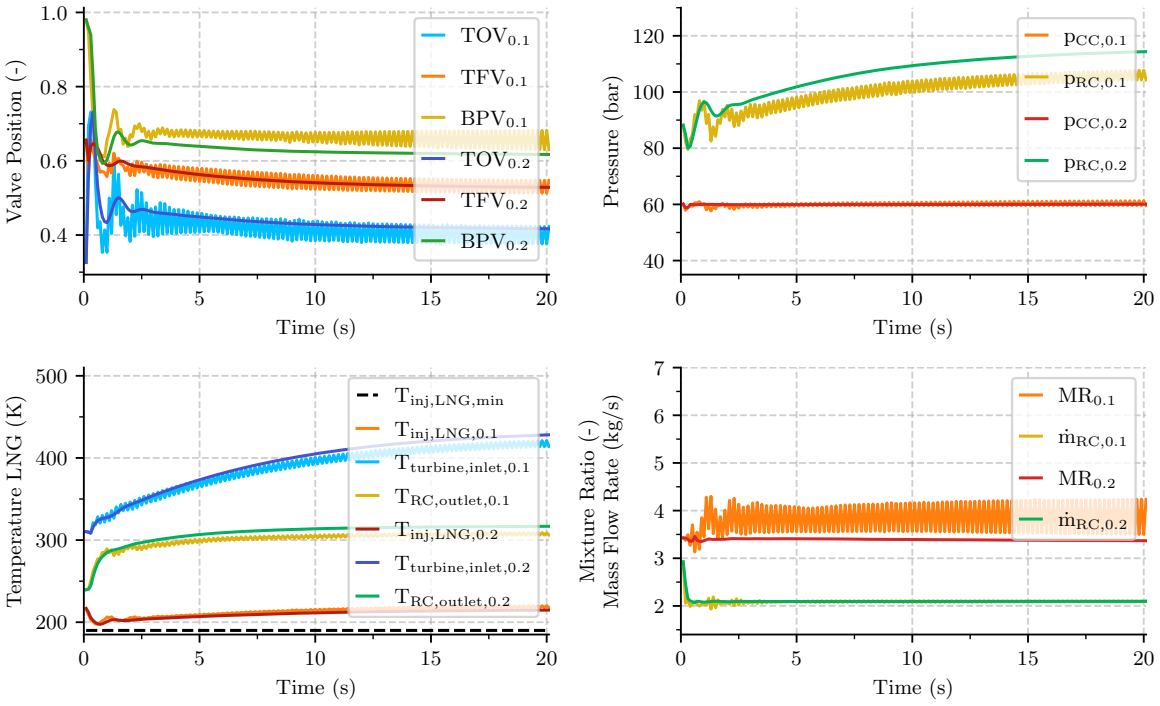


Figure 5.13.: τ Comparison when changed after Training
$\tau = 0.2$ during Training; $\tau = 0.1$ after Training

5.8. Conclusion

Combustion chamber pressure and Mixture Ratio (MR) control of LUMEN is achieved with Reinforcement Learning (RL) including operation point transition and performance optimization. LUMEN is an expander-bleed liquid rocket engine, using six controllable valves to reach its operation points.
The Soft Actor-Critic (SAC) RL algorithm is used for training the RL agent. An EcosimPro model, the simulation model, is converted into a deck, which supplies the values of input and output variables as the environment for the RL model. The robustness of the training is examined regarding changing the initial state, the impact of sensor noise, and the impact of different valve characteristics.
Combustion chamber pressure and MR control of LUMEN are achieved at different cooling channel mass flow rates. Using the Turbine Oxidizer Valve (TOV), Turbine Fuel Valve (TFV), Mixer Control Valve (XCV) and Fuel Control Valve (FCV) as adjustable valves, mass flow rates of $1.4\,\mathrm{kg\,s^{-1}}$ to $2.5\,\mathrm{kg\,s^{-1}}$ are reachable. Including the Bypass Valve (BPV) as an adjustable valve, even higher cooling channel mass flow rates are achievable, while meeting combustion chamber pressure and MR targets.
To avoid multiple solutions for one operation point, a cooling channel pressure target is implemented. A strict cooling channel pressure target value prohibits the most efficient engine operation. Engine throttling and operation point transition is performed well. Combustion chamber pressure reaching from 40 bar to 70 bar can be reached, transitioning from OP8 (80 bar, 3.8). To avoid undesired combustion chamber pressure and MR peaks during operation point transition, a reward function to suppress abrupt changes can be implemented. Abrupt variable value changes might be avoided by defining a time frame in which the operation point transition is to be made, to avoid sudden peaks as the agent is trying to maximize its reward as quickly as possible.
Low combustion chamber pressure (35 bar) cannot be reached, setting Oxidizer Combustion Valve (OCV) fully open and BPV fully closed. The output cooling channel pressure cannot be reduced without adjusting BPV and OCV to lower the combustion chamber pressure.
For optimum performance, an additional reward function to minimize the bleed mass flow rate is implemented. The new reward function minimizes the fuel dump through BPV and the turbines. It is able to reduce the total bleed mass flow rate by up to 32 %, which reduces the total LNG usage by up to 16 %. LUMENs fuel consumption is reduced and engine efficiency increased.
The robustness of the training is tested by changing the initial states, implementing a sensor noise, and changing the valve parameter τ after training. The agent can meet all targets when starting from different combustion chamber pressures than initially trained. However, it is also able to recover from starting points, which are not realistic. This shows a sim-to-real gap in the model. The sim-to-real gap has to be closed before the agent can be applied in the real engine, as unrealistic states can cause engine failure and component damage. In addition, a neural network can be implemented to determine the combustion chamber wall temperature and improve the simulation model's accuracy [62].
Adding random Gaussian sensor noise causes fluctuations. The oscillation grows as the

sensor noise increases. Even though the sensor noise has an impact on the valve position and thus all other variables, the agent can approximately follow the path of the setting without noise and hold the average combustion chamber pressure, MR and cooling channel mass flow rate at its target value.
Changing system dynamics after the training might cause problems. In this thesis the valve speed τ is changed, which causes valve fluctuation and thus system variable fluctuation. As the agent trains and learns a certain model it has difficulties adjusting to new settings. Lowering the time constant τ causes quicker valve position change, than the agent has trained for. Hence, system variables change faster. The agent tries to counteract, which results in valve fluctuation. Lowering τ causes decreasing valve response, which does not result in problems at meeting the target points. The problem of changing system dynamics can in general be addressed by using domain randomization during the training [60].

6. Summary and Outlook

Closed-loop liquid rocket engine control is a complex problem, which can be tackled using Reinforcement Learning (RL). Optimal engine performance is indispensable for reusable engines. The ability of engine throttling for maneuvers, start and landing is of high importance, which requires accurate engine control. [1]
In this thesis valve sequences for operation point transition for the Liquid Upper-stage deMonstrator ENgine (LUMEN) are generated and optimized. LUMEN is a expander-bleed liquid rocket engine, powered by LNG and LOX, and designed for test bench use, reaching combustion chamber pressures from 35 bar to 80 bar and Mixture Ratio (MR) from 3.0 to 3.8. It uses two (decoupled) turbopumps to feed the propellant into the combustion chamber. Regenerative cooling is used to lower the wall temperature of the combustion chamber and nozzle extension, while the fuel is heated for turbine entry. Six control valves can be used to adjust engine variables.
The Soft Actor-Critic (SAC) RL algorithm is used to train an agent to achieve optimal engine control. It strives to maximize a reward, which is given depending on the variable derivation from a preset setpoint.
Combustion chamber pressure and Mixture Ratio (MR) control of LUMEN is achieved with Reinforcement Learning, along with controlling the cooling channel mass flow rate, cooling channel pressure, the transition of operation points, and optimization of the engine regarding fuel consumption and its efficiency. For optimal engine control, a reward function to minimize the total bleed mass flow rate is introduced. This function can reduce fuel consumption and improve the engines' efficiency.
The training is robust against initial state change. However, a sim-to-real gap occurs, when the engine recovers from an unrealistic state. Implementing sensor noise causes target value oscillation. The training is not robust against system change after training, as the change of the valve time constant τ results in valve fluctuation, which causes variable oscillation.
To implement the training in the real world the sim-to-real gap has to be closed. An option is to implement constraints in the simulation model to avoid recovery from states, which are not realistic for the real world model. [55]
The simulation model's accuracy can be improved by implementing a neural network to determine the combustion chamber wall temperature, and hence enhance cooling channel, turbine inlet, and LNG injection temperature simulation. [62]
A reward function to avoid combustion chamber pressure and MR peaks during operation point transition should be implemented, as sharp transitions can damage the combustion chamber. The function could target to avoid abrupt combustion chamber pressure and MR transitions and give out rewards for smooth operation point transition within a defined time frame.

Training with the SAC algorithm might be improved by further hyper parameter tuning. Using `grid search`, parameters such as the entropy regularization coefficient and learning rate can be adjusted.
In addition, robustness can be improved using domain randomization as described in [60]. System parameters (such as the valve opening constant τ) can be randomized during the training to improve robustness against system changes after the training.

Bibliography

[1] A. de Iaco Veris. *Fundamental Concepts of Liquid-Propellant Rocket Engines*. Vol. 1. Springer International Publishing, 2019. ISBN: 978-3-030-54703-5. DOI: 10.1007/978-3-030-54704-2.

[2] K. Dresia, S. Jentzsch, G. Waxenegger-Wilfing, R. D. Santos Hahn, J. Deeken, M. Oschwald, and F. Mota. "Multidisciplinary Design Optimization of Reusable Launch Vehicles for Different Propellants and Objectives". In: *Journal of Spacecraft and Rockets* 0.0 (2020), pp. 1–13. DOI: 10.2514/1.A34944.

[3] *Falcon User's Guide*. Apr. 2020. URL: https://www.spacex.com/media/falcon_users_guide_042020.pdf. Accessed: 02.01.2021.

[4] S. Pérez Roca, J. Marzat, H. Piet-Lahanier, N. Langlois, F. Farago, M. Galeotta, and S. Gonidec. "A survey of automatic control methods for liquid-propellant rocket engines". In: *Progress in Aerospace Sciences* 107 (May 2019), pp. 63–84. DOI: 10.1016/j.paerosci.2019.03.002.

[5] C. F. Lorenzo and J. L. Musgrave. "Overview of rocket engine control". In: *AIP Conference Proceedings* 246.1 (1992), pp. 446–455. DOI: 10.1063/1.41807.

[6] D. Preclik, R. Strunz, G. Hagemann, and G. Langel. "Reusability aspects for space transportation rocket engines: Programmatic status and outlook". In: *CEAS Space Journal* 1 (Sept. 2011), pp. 71–82. DOI: 10.1007/s12567-011-0006-x.

[7] G. Waxenegger-Wilfing, K. Dresia, J. Deeken, and M. Oschwald. "Machine Learning Methods for the Design and Operation of Liquid Rocket Engines - Research Activities at the DLR Institute of Space Propulsion". In: *Space Propulsion 2020 Conference*. Mar. 2021.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (second edition)*. Second. The MIT Press, 2018.

[9] J. Kober, J. Bagnell, and J. Peters. "Reinforcement Learning in Robotics: A Survey". In: *The International Journal of Robotics Research* 32 (Sept. 2013), pp. 1238–1274. DOI: 10.1177/0278364913495721.

[10] O. Haidn. "Advanced Rocket Engines". In: *Advances on Propulsion Technology for High-Speed Aircraft* (2008), pp. 6.1–6.40.

[11] S. Pérez Roca, J. Marzat, H. Piet-Lahanier, N. Langlois, M. Galeotta, F. Farago, and S. Gonidec. "Model-based Robust Transient Control of Reusable Liquid-Propellant Rocket Engines". In: *IEEE Transactions on Aerospace and Electronic Systems* PP (July 2020), pp. 1–1. DOI: 10.1109/TAES.2020.3010668.

[12] Y. Zhang, J. Wu, M. Huang, H. Zhu, and Q. Chen. "Liquid-Propellant Rocket Engine Health-Monitoring Techniques". In: *Journal of Propulsion and Power* 14 (Sept. 1998). DOI: 10.2514/2.5327.

[13] J. Musgrave, D. Paxson, J. Litt, and W. Merrill. "A demonstration of an intelligent control system for a reusable rocket engine". In: *Advanced Earth-to-Orbit Propulsion Technology Conference* (July 1992).

[14] H. Sunakawa, A. Kurosu, K. Okita, W. Sakai, S. Maeda, and A. Ogawara. "Automatic Thrust and Mixture Ratio Control of the LE-X". In: (July 2008). DOI: 10.2514/6.2008-4666.

[15] W. Kitsche. *Operation of a Cryogenic Rocket Engine*. Vol. 2. Nov. 2010. ISBN: 978-3-642-10564-7. DOI: 10.1007/978-3-642-10565-4.

[16] C. F. Promper. "Electrically actuated regulation valves for rocket engines". In: *9th European Space Mechanisms and Tribology Symposium* (Sept. 2001), pp. 183–189.

[17] A. Bhatia. *Control Valve Basics - Sizing & Selection*. Createspace Independent Pub, 2014. ISBN: 9781502841070.

[18] *ESPPS User Manual*. English. Version 3.3.0. EcosimPro Modelling and Simulation Software, Empresarios Agrupados Internacional S.A. Feb. 2019.

[19] S. B. Reddy. *Short Notes on Different Valve Types*. 2019. URL: https://instrumentationtools.com/short-notes-different-valve-types/. Accessed: 02.10.2020.

[20] E. Betts and R. Frederick. "A Historical Systems Study of Liquid Rocket Engine Throttling Capabilities". In: July 2010. ISBN: 978-1-60086-958-7. DOI: 10.2514/6.2010-6541.

[21] D. Bradley and K. Hooser. "Space Shuttle Main Engine - The Relentless Pursuit of Improvement". In: *AIAA Space 2011 Conference & Exposition*. Sept. 2011. ISBN: 978-1-60086-953-2. DOI: 10.2514/6.2011-7159.

[22] P. F. Seitz and R. F. Searle. "Space Shuttle Main Engine Control System". In: *SAE Technical Paper*. SAE International, Feb. 1973. DOI: 10.4271/730927.

[23] P. Brossel, P. Caisso, M. Illig, and T. Margat. "Development Status of the Vulcain 2 Engine". In: *30th Joint Propulsion Conference and Exhibit*. July 2002. ISBN: 978-1-62410-115-1. DOI: 10.2514/6.2002-3840.

[24] P. Alliot, J.-F. Delange, V. Korver, J.-M. Sannino, A. Lekeux, and B. Vieille. "VINCI ® , the european reference for ariane 6 upper stage cryogenic propulsive system". In: *Progress in Propulsion Physics (Volume 11)*. Jan. 2019, pp. 481–494. DOI: 10.1051/eucass/201911481.

[25] A. Iannetti, N. Girard, D. Tchou-kien, C. Bonhomme, and E. Ravier N.and Edeline. "Prometheus, a LOX/LCH4 reusable rocket engine". In: *7th European Conference for Aeronautics and Space Science (EUCASS)* (2017). DOI: 10.13009/EUCASS2017-537.

[26] J. Hardi, J. Martin, M. Son, W. Armbruster, J. Deeken, D. Suslov, and M. Oschwald. "Combustion Stability Characteristics of a sub-scale LOX/LNG Rocket Thrust Chamber". In: *Aerospace Europe Conference 2020 (AEC 2020)*. Feb. 2020.

[27] T. Traudt, T. Mason, J. Deeken, M. Oschwald, S. Schlechtriem, R. H. dos Santos Hahn, and C. Mader. "LUMEN Turbopump -Design and Manufacturing of the LUMEN LOX and LNG Turbopump components". In: *International Symposium on Space Technology and Science (ISTS)*. June 2019.

[28] G. Waxenegger-Wilfing, K. Dresia, J. Deeken, and M. Oschwald. "A Reinforcement Learning Approach for Transient Control of Liquid Rocket Engines". In: *IEEE Transactions on Aerospace and Electronic Systems* (2021). DOI: 10.1109/TAES.2021.3074134.

[29] T. Traudt, J. Deeken, M. Oschwald, and S. Schlechtriem. "Liquid Upper Stage Demonstrator Engine (LUMEN): Status of the Project". In: *70th International Astronautical Congress (IAC)*. 2019.

[30] J. Haemisch, D. Suslov, G. Waxenegger-Wilfing, K. Dresia, and M. Oschwald. "LUMEN - Design of the Regenerative Cooling System for an Expander Bleed Cycle Engine Using Methane". In: *Space Propulsion 2020+1 Conference (Virtual Event)*. 2021.

[31] K. Dresia, G. Waxenegger-Wilfing, R. H. dos Santos Hahn, J. Deeken, and M. Oschwald. "Nonlinear Control of an Expander-Bleed Rocket Engine using Reinforcement Learning". In: *Space Propulsion 2020+1 Conference (Virtual Event)*. Mar. 2021.

[32] J. Deeken, G. Waxenegger-Wilfing, and R. H. dos Santos Hahn. "LUMEN Technical Specification DEMO (TS-DEMO) (restricted document)". In: (Mar. 2020).

[33] J. Deeken, G. Waxenegger-Wilfing, M. Oschwald, and S. Schlechtriem. "LUMEN Demonstrator Project Overview". In: *Space Propulsion 2020+1 Conference (Virtual Event)*. Mar. 2021.

[34] A. Heintz. *Thermodynamik der Mischungen und Mischphasengleichgewichte*. Springer Berlin Heidelberg, 2017. ISBN: 978-3-662-49924-5. DOI: 10.1007/978-3-662-49924-5_1.

[35] *Complete Reference Manual*. English. Version 6.2.0. EcosimPro Modelling and Simulation Software, Empresarios Agrupados Internacional S.A. 2020.

[36] T. Traudt, G. Waxenegger-Wilfing, R. H. dos Santos Hahn, B. Wagner, and J. Deeken. "An Overview on the Turbopump Roadmap for the LUMEN Demonstrator Engine and on the new Turbine Test Facility". In: *68th International Astronautical Congress (IAC)*. Sept. 2017.

[37] A. I. Edelman. *Propellant Valves of Liquid-Propellant Rocket Engines*. Foreign Technology Division, Air Force Systems Command (U.S. Air Force), 1972.

[38] J. Moral, R. Vara, J. Steelant, and M. Rosa. "ESPSS Simulation Platform". In: *Space Propulsion 2010 Conference*. May 2010.

[39] *Thermal Libaray*. English. Version 3.5.2. EcosimPro Modelling and Simulation Software, Empresarios Agrupados Internacional S.A. 2016.

[40] Y. Li. "Deep Reinforcement Learning: An Overview." In: *Proceedings of SAI Intelligent Systems Conference* (2017). DOI: 10.1007/978-3-319-56991-8_32.

[41] E. Even-Dar, S. M. Kakade, and Y. Mansour. "Experts in a Markov Decision Process". In: *Proceedings of the 17th International Conference on Neural Information Processing Systems*. NIPS'04. Vancouver, British Columbia, Canada: MIT Press, 2004, pp. 401–408.

[42] I. Galatzer-Levy, K. Ruggles, and Z. Chen. "Data Science in the Research Domain Criteria Era: Relevance of Machine Learning to the Study of Stress Pathology, Recovery, and Resilience". In: *Chronic Stress* 2 (Jan. 2018). DOI: 10.1177/2470547017747553.

[43] OpenAI. *Intrpduction to RL*. URL: https://spinningup.openai.com/en/latest/index.html. Accessed: 04.01.2021.

[44] J. Sharma, P.-A. Andersen, O.-C. Granmo, and M. Goodwin. "Deep Q-Learning with Q-Matrix Transfer Learning for Novel Fire Evacuation Environment". In: *IEEE Transactions on Systems, Man, and Cybernetics* (2019). DOI: 10.1109/TSMC.2020.2967936.

[45] S. Heinz. *Einführung in Reinforcement Learning – wenn Maschinen wie Menschen lernen*. STATWORX. 2018. URL: https://www.statworx.com/at/blog/einfuehrung-in-reinforcement-learning-wenn-maschinen-wie-menschen-lernen/. Accessed: 08.03.2021.

[46] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (first edition)*. Bradford Books, MIT Press, Feb. 1998.

[47] K. Hovell and S. Ulrich. "On Deep Reinforcement Learning for Spacecraft Guidance". In: *AIAA Scitech 2020 Forum*. Jan. 2020. DOI: 10.2514/6.2020-1600.

[48] J. Schulman, P. Abbeel, and X. Chen. "Equivalence Between Policy Gradients and Soft Q-Learning". In: *CoRR* abs/1704.06440 (2017).

[49] H. Jomaa, J. Grabocka, and L. Schmidt-Thieme. *Hyp-RL : Hyperparameter Optimization by Reinforcement Learning*. June 2019.

[50] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning." In: *4th International Conference on Learning Representations*. 2016.

[51] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. 2014, pp. 387–395.

[52] S. Fujimoto, H. van Hoof, and D. Meger. "Addressing function approximation error in actor-critic methods". In: *CoRR* abs/1802.09477 (2018).

[53] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. "Soft Actor-Critic Algorithms and Applications". In: *CoRR* abs/1812.05905 (2018).

[54] OpenAI. *Soft Actor-Critic*. URL: https://spinningup.openai.com/en/latest/algorithms/sac.html?highlight=actor-critic. Accessed: 04.01.2021.

[55] H. Dong, Z. Ding, and S. Zhang. *Deep Reinforcement Learning Fundamentals, Research and Applications: Fundamentals, Research and Applications*. Jan. 2020. ISBN: 978-981-15-4094-3. DOI: 10.1007/978-981-15-4095-0.

[56] G. Paolo, A. Laflaquière, A. Coninx, and S. Doncieux. "Unsupervised Learning and Exploration of Reachable Outcome Space". In: *IEEE International Conference on Robotics and Automation (ICRA) 2020*. May 2020. DOI: 10.1109/ICRA40945.2020.9196819.

[57] E. Liang, Z. Wu, M. Luo, S. Mika, and I. Stoica. *RLlib Flow: Distributed Reinforcement Learning is a Dataflow Problem*. 2021.

[58] J. Hare. "Dealing with Sparse Rewards in Reinforcement Learning". In: *CoRR* abs/1910.09281 (2019).

[59] G. Waxenegger-Wilfing, K. Dresia, M. Oschwald, and K. Schilling. "Hardware-In-The-Loop Tests of Complex Control Software for Rocket Propulsion Systems". In: *71st International Aeronautical Congress*. Oct. 2020.

[60] T. Hörger. "Reinforcement Learning Framework for Optimal Control of Orbital Propulsion considering Systems Robustness and Operating Limitations". Jan. 2021. DOI: 10.13140/RG.2.2.22591.23200.

[61] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz. "Parameter Space Noise for Exploration". In: *CoRR* abs/1706.01905 (2017).

[62] G. Waxenegger-Wilfing, K. Dresia, J. Deeken, and M. Oschwald. "Heat Transfer Prediction for Methane in Regenerative Cooling Channels with Neural Networks". In: *Journal of Thermophysics and Heat Transfer* (Jan. 2020). DOI: 10.2514/1.T5865.

A. Appendix

A.1. SAC Parameter Configuration

Table A.1.: Parameter Configuration of SAC-Algorithm applied for all Calculations of this Thesis

Parameter	Value
num_workers	1
num_envs_per_worker	1
create_env_on_driver	False
rollout_fragment_length	1
batch_mode	truncate_episodes
num_gpus	0
train_batch_size	256
model	MODEL_DEFAULTS
optimizer	
gamma	0.9
horizon	200
soft_horizon	False
no_done_at_end	True
env_config	LumenEnv
env	None
normalize_actions	True
clip_rewards	None
clip_actions	True
preprocessor_pref	deepmind
lr	0.0001
monitor	False
log_level	WARN
callbacks	DefaultCallbacks
ignore_worker_failures	False
log_sys_usage	True
fake_sampler	False
framework	torch
eager_tracing	False

continued on next page

Table A.1 – continued from previous page

Parameter	Value
explore	True
exploration_config	
type	StochasticSampling
evaluation_interval	1
evaluation_num_episodes	10
in_evaluation	False
evaluation_config	
evaluation_num_workers	0
custom_eval_function	None
sample_async	False
_use_trajectory_view_api	False
observation_filter	NoFilter
synchronize_filters	True
tf_sessions_args	
intra_op_parallelism_threads	2
inter_op_parallellism_threads	2
gpu_options	
allow_growth	True
log_device_placement	False
device_count	
CPU	1
allow_soft_placement	True
local_tf_session_args	
intra_op_parallelism_threads	8
inter_op_parallelism_threads	8
compress_observations	False
collect_metrics_timeout	180
metrics_smoothing_episodes	100
remote_worker_envs	False
remote_env_batch_wait_ms	0
min_iter_time_s	0
timesteps_per_iteration	5000
seed	None
extra_python_environs_for_driver	
extra_python_environs_for_worer	
num_cpus_per_worker	1
num_gpus_per_worker	0
custom_resources_per_worker	
num_cpus_for_driver	1

continued on next page

Table A.1 – continued from previous page

Parameter	Value
memory	0
object_store_memory	0
memory_per_worker	0
object_store_memory_per_worker	0
input	sampler
input_evaluation	[is, wis]
postprocess_inputs	False
shuffle_buffer_size	0
output	None
output_compress_columns	[obs, new_obs]
output_mac_file_size	
muliafent	
policies	
policy_mapping_fn	None
policies_to_train	None
observation_fn	None
replay_mode	independent
logger_config	None
replay_sequence	1
twin_q	True
use_state_preprocessor	False
Q_model	
fcnet_activation	relu
fcnet_hiddens	[256, 256]
policy_model	
fcnet_activation	relu
fcnet_hiddens	[256, 256]
tau	5e-3
initial_alpha	1.0
target_entropy	auto
n_step	1
buffer_size	100000
prioritized_replay	False
prioritized_replay_alpha	0.6
prioritized_replay_beta	0.4
prioritized_replay_eps	1e-6
prioritized_replay_beta_annealing_timesteps	20000
final_prioritized_replay_beta	0.4
compress_observation	False

continued on next page

Table A.1 – continued from previous page

Parameter	Value
training_intensity	None
optimization	
actor_learning_rate	3e-4
critic_learning_rate	3e-4
entropy_learning_rate	3e-4
grad_clip	None
learning_starts	256
target_network_update_freq	0
worker_side_prioritization	False
min_iter_time	1
_deterministic_loss	False
_use_beta_distribution	False

A.2. Checkpoint Comparison Temperature Constraint

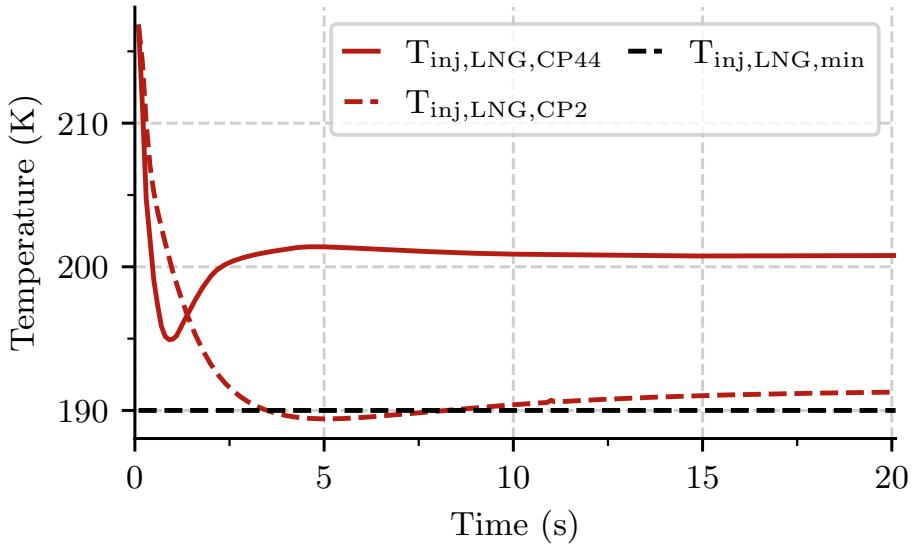


Figure A.1.: Checkpoint Comparison Temperature Constraint: (–) CP 2, (- -) CP 44

A.3. System Change after Training

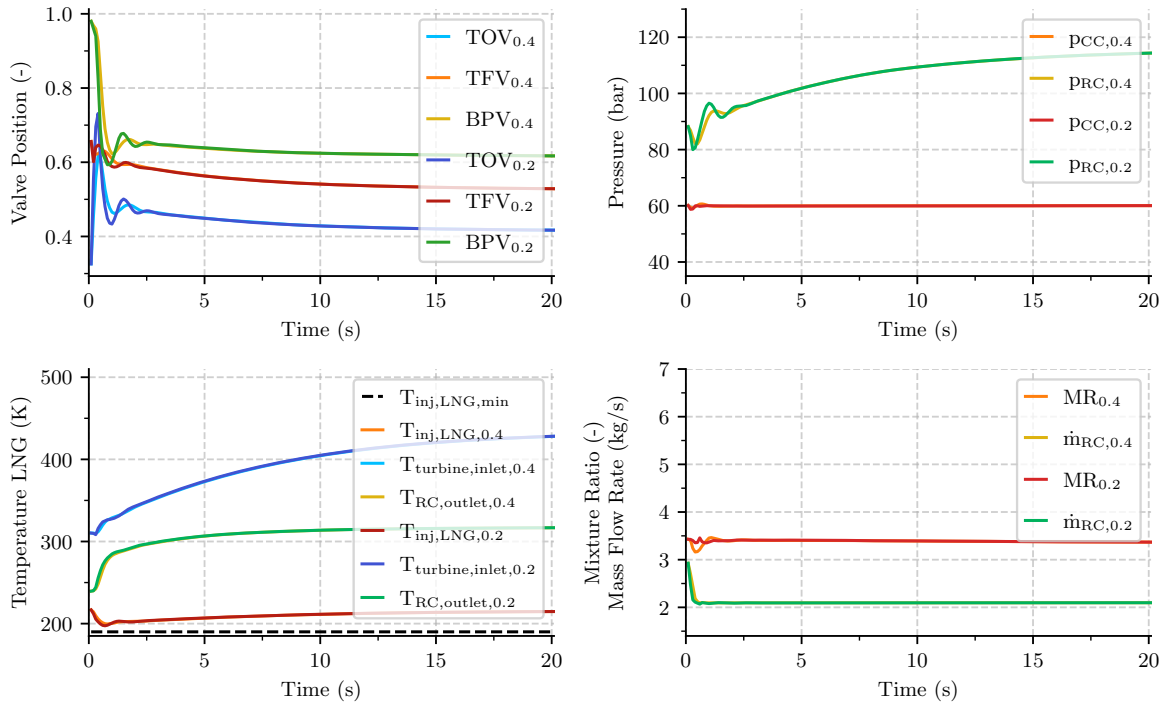


Figure A.2.: τ Comparison when changed after Training
$\tau = 0.2$ during Training; $\tau = 0.4$ after Training